VISUAL ANALYSIS OF HIGH-DIMENSIONAL SPACES

Von der Carl-Friedrich-Gauß Fakultät

der Technischen Universität Carolo-Wilhelmina zu Braunschweig

zur Erlangung des Grades einer

Doktoringenieurin (Dr.-Ing.)

genehmigte

Dissertation

von	Georgia Priscylla Cesar de Albuquerque Richers
geboren am	25. September 1979
in	Recife-PE, Brasilien

Eingereicht am: 6. Februar 2014

Disputation am: 9. Mai 2014

1. Referent: Prof. Dr.-Ing. Marcus Magnor

2. Referent: Prof. Dr. Daniel Keim

(2014)

Georgia P. C. de Albuquerque Richers: Visual Analysis of High-Dimensional Spaces

Die Edition Octopus erscheint im Verlagshaus
Monsenstein und Vannerdat OHG Münster
www.edition-octopus.de

Satz und Umschlag: G. P. C de Albuquerque Richers
Illustrationen: G. P. C de Albuquerque Richers
Druck und Bindung: MV-Verlag

ISBN

Abstract

The visual exploration and analysis of high-dimensional data sets commonly requires projecting the data into lower-dimensional representations. The number of possible representations grows rapidly with the number of dimensions, and manual exploration quickly becomes ineffective or even infeasible. In this thesis I present automatic algorithms to compute visual quality metrics and show different situations where they can be used to support the analysis of high-dimensional data sets. The proposed methods can be applied to different specific user tasks and can be combined with established visualization techniques to sort or select projections of the data based on their information-bearing content. These approaches can effectively ease the task of finding truly useful visualizations and potentially speed up the data exploration task. Additionally, I present a framework designed to generate synthetic data, where users can interactively create and navigate through high dimensional data sets.

Kurzfassung

Die visuelle Untersuchung und Analyse von hochdimensionalen Datensätzen erfordert in der Regel die Projektion der Daten in niedrig-dimensionale Repräsentationen. Die Anzahl möglicher Repräsentationen wächst dabei rapide mit der Anzahl der Dimensionen, was eine manuelle Untersuchung ineffektiv oder sogar unmöglich macht. In dieser Arbeit stelle ich Algorithmen zur Berechnung visueller Metriken vor und zeige verschiedene Situationen, in welchen diese Verwendung finden können, um die Untersuchung von hochdimensionalen Datensätzen zu unterstützen. Durch Sortierung und Auswahl der Projektionen, basiered auf ihrem Informationsgehalt, können die vorgeschlagenen Metriken für verschiedene spezifische *User tasks* und zusammen mit etablierte Visualisierungstechniken angewendet werden. Dieses Vorgehen kann die Suche nach aussagekräftigen Visualisierungen effektiv vereinfachen und so die Datenuntersuchung und Analyse beschleunigen. Ich präsentiere ausserdem ein Framework, das entworfen ist, um synthetische Daten zu generieren, bei dem der Benutzer interaktiv neue hochdimensionale Datensätze erzeugen und untersuchen kann.

Summary

Modern visualization methods and visual analytics approaches are needed to cope with very high-dimensional data. The large number of possible projections for existing visualization techniques, which usually grow quadratically or even exponentially with the number of dimensions, urges the necessity to employ automatic approaches to reduce the number of dimension or to select the best projections, based on their information-bearing content. In this thesis several approaches to support the exploration of such high-dimensional data sets are presented.

The first part of this thesis introduces different quality metrics that can successfully be applied to different specified user tasks and established visualization techniques, including scatterplot matrices, parallel coordinates, radviz and pixel-based visualizations. Furthermore, as an extension to this set of quality metrics, a perception-based quality metric that is able to mimic user opinion about the quality of projections of the data is presented. These metrics can be very useful to rank and select information-bearing projections of very high dimensional data, e.g. when the visual exploration of all possible projections becomes infeasible.

Following the introduction of quality metrics, new visualization approaches that are able to support the visual exploration task of multivariate data sets are described. Specially, a parallel coordinates matrix, in analogy to the well-known scatterplot matrix; a class-based scatterplot matrix that aims at finding good projections for each class pair; an importance-aware algorithm to sort the dimensions of scatterplot and parallel coordinates matrices; and a content-aware color mapping algorithm are presented.

Finally, an interactive framework designed to generate high-dimensional data sets is introduced. Data generation is driven by statistical distributions based on a few user-defined parameters. First, a default data set is created according to given input, then structures and trends are included in selected dimensions and orthogonal projection planes. The framework supports the creation of complex non-orthogonal trends and classified data sets. It can successfully be used to create synthetic data sets simulating important trends as multidimensional clusters, correlations, and outliers.

Zusammenfassung

Moderne Visualisierungsmethoden und Visual-Analysis-Verfahren sind notwendig, um hochdimensionale Daten zu verarbeiten. Die beträchtliche Anzahl möglicher Projektionen für die existierenden Visualisierungstechniken, welche in der Regel quadratisch oder sogar exponentiell mit der Anzahl der Dimensionen wachsen, machen es nötig automatisierte Verfahren anzuwenden, um die Anzahl der Dimensionen zu reduzieren, oder um die besten Projektionen, basierend auf ihrem informationellen Inhalt, auszuwählen. In dieser Arbeit werden mehrere Herangehensweisen vorgestellt, um die Untersuchung solcher hochdimensionaler Datensätze zu unterstützen.

Der erste Teil dieser Arbeit führt unterschiedliche Qualitätsmetriken ein, welche für mehrere spezifische Anwendungsfälle und etablierte Visualisierungstechniken, wie Scatterplot-Matrizen, Parallel-Koordinaten, sowie Radviz und pixel-basierten Visualisierungen, erfolgreich eingesetzt werden können. Desweiteren, um den Satz der Qualitätsmetriken zu komplettieren, wird eine wahrnehmungs-basierte Qualitätsmetrik vorgestellt, die es ermöglicht das Benutzerurteil über die Qualität von Datenprojektionen zu imitieren. Dieser Satz von Metriken ist sehr nützlich, um informationstragende Projektionen von hochdimensionalen Datensätzen auszuwählen und einzuordnen, wenn die visuelle Untersuchung aller möglichen Projektionen undurchführbar ist.

Nach der Einführung der Qualtitätsmetriken werden Visualisierungsverfahren beschrieben, die die visuelle Untersuchung von multivariaten Da-

tensätzen, basierend auf diesen Metriken, unterstützen können. Dies beinhaltet eine Parallelkoordinaten-Matrix, in Analogie zu den wohlbekannten Scatterplot-Matrizen; eine klassenbasierte Scatterplot-Matrix, die eine gute Projektion aller Klassenpaare anvisiert; einen Algorithmus zum Sortieren der Dimensionen von Scatterplot- und Parallelkoordinaten-Matrizen nach Wichtigkeit; und einen content-sensitiven Color-Mapping-Algorithmus.

Abschließend wird ein Framework zur Generierung von hochdimensionalen Datensätzen vorgestellt, bei dem der Benutzer interaktiv multidimensionale Datensätze, mit Hilfe eines geeigneten Benutzerinterfaces, erstellen und untersuchen kann. Die Datenerstellung wird durch statistische Verteilungen, basierend auf wenigen benutzerdefinierten Parametern, gesteuert. Zunächst wird ein grundlegender Datensatz, entsprechend gegebener Eingaben, erstellt, woraufhin Strukturen und Trends in ausgewählten Dimensionen und orthogonalen Projektionsebenen eingefügt werden. Desweiteren unterstützt das Framework die Erstellung komplexer nicht-orthogonaler Trends und klassifizierter Datensätze. Es kann erfolgreich eingesetzt werden, um synthetische Datensätze zu erstellen, welche wichtige Trends, wie multidimensionale Cluster, Korrelationen und Ausreißer, zu simulieren.

To my parents,
and to Oliver and Isabella

Acknowledgements

First of all I would like to thank my supervisor Marcus Magnor for believing in me, for his steady support and all the valuable and constructive comments on this research work. I would like to express my very great appreciation to present and former colleagues of the Computer Graphics Lab at TU Braunschweig. Thanks Timo Stich, Cristian Linz, Anita Sellent, Chistian Lipski, Kai Berger, Stephan Wenger, Lorenz Rogge, Felix Klose, Kai Ruhl, Maryam Mustafa, Pablo Bauszat, Benjamin Hell, Michael Stengel, Stefan Guthe, Thomas Neumann, and others for creating a great and pleasant working atmosphere, for the insightful comments and for enjoyable lunch breaks. My special thanks goes to Martin Eisemann for many constructive suggestions and encouragements. Thanks Anja Franzmeier for supporting all of us in the administrative work and Carsten Golze for maintaining all computers and solving many technical problems. Further thanks go also to my student assistants Thomas Löwe and Torben Rodermund.

Part of this thesis was developed in collaboration with other groups. In this context I wish to thank my collaboration partners Dirk Lehmann, Andrada Tatu, Holger Theisel and Daniel Keim for the valuable discussions and inputs for this work. I would also like to express my gratitude to the financial support received from the German Research Society (DFG) project DFG MA2555/6-1 within the strategic research initiative on Scalable Visual Analytics (SPP 1335).

Finally, I would like to thank my family for the support they provided me through my entire life. A very special thanks goes to Oliver for his love and warm encouragement.

Contributions of the Author

Clarification of my individual contributions to the publications that describe parts of my thesis; The papers are ordered according to the structure of this thesis.

1. Andrada Tatu, Georgia Albuquerque, Martin Eisemann, Jörn Schneidewind, Holger Theisel, Marcus Magnor, and Daniel Keim. **Combining automated analysis and visualization techniques for effective exploration of high dimensional data**. In Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (IEEE VAST), pages 59–66, 2009

   This paper was the result of a cooperation between the Institut für Computergraphik *at TU Braunschweig, the Data Analysis and Visualization group at the University of Konstanz, and the Department of Simulation and Graphics at the University of Magdeburg. It was selected as one of the three best papers in the VAST'09 conference and won the SPP Collaboration Award in the DFG priority program on Scalable Visual Analytics (SPP 1335). I developed the ideas and implemented two of the three quality metrics presented in the paper for scatterplots, namely the* Rotating Variance Measure *and the* Class Density measure*, and all three metrics for parallel coordinates (*Hough Space Measure*,* Similarity Measure *and* Overlap Measure*). These contributions are part of Chapter 2. M. Eisemann and I discussed different ideas for all quality metrics and he gave me advice for the implementation. A. Tatu implemented another quality metric for scatterplots (*Histogram Density Measure*), which is not part of this thesis, and provided the respective description and evaluation in the paper. A. Tatu, M. Eisemann and I worked closely together while writing the paper. M. Magnor, D. Keim, H. Theisel and J. Schneidewind guided the project with many suggestions and gave advice concerning ideas and content of the paper.*

2. Andrada Tatu, Georgia Albuquerque, Martin Eisemann, Peter Bak, Holger Theisel, Marcus Magnor, and Daniel Keim. **Automated analytical methods to support visual exploration of high-dimensional data**. IEEE Transactions on Visualization and Computer Graphics (TVCG), 17(5):584–597, 2011.

   Since Paper 1 was selected as one of the best of the VAST'09 conference, the authors were invited to extend on it in TVCG, resulting in this article. For this extension, I implemented and described an additional quality metric for

scatterplots called Class Separation Metric *and conducted new experiments and results for this and other existing metrics. These contributions are part of Chapter 2. A. Tatu organized the journal article, generated new use-cases for the experiments and also evaluated the existing metrics. The new experiments generated by her are not included in my thesis. P. Bak helped in the structure of the journal and oriented the new evaluation for the metrics. D. Keim, M. Eisemann, M. Magnor and H. Theisel gave suggestions and advice concerning ideas and contents of the paper.*

3. Georgia Albuquerque, Martin Eisemann, Dirk J. Lehmann, Holger Theisel, and Marcus Magnor. **Improving the visual analysis of high-dimensional data sets using quality measures**. In Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (IEEE VAST), 2010.

   This paper was joint work with our project partners at the Department of Simulation and Graphics of the University of Magdeburg. My contribution to this paper were the ideas for the quality metrics for radviz and table lens, the implementation and evaluation of the quality metrics for radviz, and writing the paper. These contributions are part of Chapter 2. M. Eisemann and I implemented and evaluated together the quality metric for jigsaw maps. D. J. Lehmann implemented and evaluated the quality metrics for table lens. These quality metrics for table lens are not included in my thesis because I was not involved in the implementation. M. Magnor and H. Theisel supervised the project with suggestions and advice.

4. Georgia Albuquerque, Martin Eisemann, and Marcus Magnor. **Perception-based visual quality measures**. In Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (IEEE VAST) 2011, pages 13–20, 2011.

   I was responsible for the ideas, implementing most algorithms included in this work, conducting the user studies, evaluating the perception-based metric and writing the paper. These contributions are part of Chapter 3. M. Eisemann supported me with advice and suggestions for the implementation and evaluation of the algorithms. M. Magnor oversaw the project.

5. Georgia Albuquerque, Martin Eisemann, Dirk. J. Lehmann, Holger Theisel, and Marcus Magnor. **Quality-based visualization matrices**. In Proceedings of Vision, Modeling and Visualization (VMV) 2009, pages 341–349, 2009.

This paper was joint work with our project partners at the Department of Simulation and Graphics of the University of Magdeburg. My contribution to this work was the idea, implementation and evaluation of the class-based scatterplot matrix and the parallel coordinate matrix. These contributions are part of Chapter 4. M. Eisemann gave advice for the implementation and evaluation of the methods. D. J. Lehmann and I implemented and evaluated the reordering algorithm for scatterplot matrices. M. Magnor and H. Theisel supervised the project.

6. Martin Eisemann, Georgia Albuquerque, and Marcus Magnor. **Data driven color mapping**. In Proceedings of the International Workshop on Visual Analytics 2011 (EuroVA), pages 5-8, 2011.

   The ideas presented in this paper are a joint contribution of M. Eisemann and me. I was responsible for the idea, implementation, evaluation and the description of the color mapping algorithm in the paper. These contributions are part of Chapter 4. M. Eisemann implemented the interpolation algorithm and we wrote the paper together. M. Magnor supervised the project.

7. Georgia Albuquerque, Thomas Löwe, and Marcus Magnor. **Synthetic generation of high-dimensional data sets**. IEEE Transactions on Visualization and Computer Graphics (TVCG, Proc. Visualization / InfoVis), 17(12):2317–2324, 2011.

   In this work I was responsible for the ideas presented in this paper and writing the paper. These contributions are part of Chapter 5. T. Löwe and I developed the presented algorithms for the synthetic generation of multidimensional data sets and evaluated the methods in close cooperation. M. Magnor supervised the project.

Other co-authored publications that are not included in this thesis:

1. Dirk. J. Lehmann, Georgia Albuquerque, Martin Eisemann, Marcus Magnor, and Holger Theisel. **Selecting Coherent and Relevant Plots in Large Scatterplot Matrices**. Computer Graphics Forum, vol. 31, no. 6, pp. 1895–1908, 2012.

2. Dirk. J. Lehmann, Georgia Albuquerque, Martin Eisemann, Andrada Tatu, Heidrun Schumann, Marcus Magnor, and Holger Theisel. **Visualisierung**

und Analyse multidimensionaler Datensätze. Informatik-Spektrum, vol. 33, no. 5, pp. 589–600, 2010.

Contents

CONTENTS

Chapter 1

Introduction

1.1 Motivation

Information is the foundation of any decision making, be it in natural science, medical and health care, economy or society. Due to the technological progress over the last years, today's scientific and commercial applications are capable of generating, storing, and processing large and complex sets of data. Imagine a data set that describes a set of objects, and the high dimensionality is a direct result of trying to describe the objects through a collection of features. A practical example is a set of cars consisting of multiple samples (records), each sample having multiple attributes, where each attribute describes a feature of the car. These attributes may include: name, origin, horsepower, weight, acceleration, cylinders, model year, etc. If we now join all car models that have been produced in recent years and how many possible features can be used to describe or even distinguish all those models, we would have a very large and complex data set. Filtering and extracting relevant information from these masses of data is becoming more and more difficult since the complexity and volume of such data sets is steadily increasing.

The visualization of large and complex information spaces typically involves mapping high-dimensional data to lower-dimensional visual representations, such that properties, relationships and functional dependencies in the data may be revealed. The challenge for the analyst is to find an insightful mapping, while the dimensionality of

the data, and consequently the number of possible mappings increases. Using scatterplots, for example, there are $\frac{n^2-n}{2}$ possible projections to visualize the entire the data set.

Numerous expressive and effective low-dimensional visualization approaches for high-dimensional data sets have been proposed in the past, such as scatterplot matrices (SPLOM), parallel coordinates, hyper-slices, dense pixel displays and geometrically transformed displays [KAS04]. However, one of the problems of these visualization methods alone is that they do not scale well when the number of dimensions grows. Finding information-bearing and user-interpretable visual representations remains a difficult task because the number of possible representations can be so large that a manual exploration is not feasible. Therefore, more effective visual exploration techniques are necessary that incorporate automated analysis components to reduce complexity and to guide the user during the interactive exploration process.

Automated approaches from well-established areas such as data mining or computer vision can also be used to identify hidden patterns in the data. A very successful data mining example is "Market Basket Research" where the main goal is to find associations between supermarket basket items in a purchase transaction [AIS93]. This information can be very useful for typical business decisions, such as how to organize the products in the supermarket sections or to decide which products should be put on sale together. Such approaches can are useful to solve a lot of different problems in various application areas, e.g. search for associations between medical diagnosis tests that can be used to find out which medical test could be replaced by others [AMS97]. In Computer vision, there are many algorithms that can be used to find patterns by analyzing image content. Face and Fingerprint recognition are very established application examples. The question arises whether these algorithms can be combined to visualization techniques to find hidden patterns in the data as an alternative to traditional methods.

Visual analytics combines the analytic power of computers and the ability of human analysts to create new exploration possibilities and to allow individuals to take control of the analytical process [KKEM10]. Frequently, solving real-life problems

requires searching for information, e.g. trends or clusters, in multidimensional data sets. An important goal of visual analytics approaches is to generate representations that best show such information contained in high-dimensional data. This information is commonly identified through correlations or clusters that are found in the data set.

This thesis presents different approaches to support the search for information-bearing representations of high-dimensional data sets. The methods are motivated by the interdisciplinary essence of visual analytics to combine the power and experience of automated pattern analysis and pattern recognition with the interactive visual exploration of complex data sets. Until now, little research has been conducted to investigated the idea of applying image analysis techniques to support the exploration of non-visual data [SSK06]. Using image analysis techniques we can capitalize on some inherent aspects, including the natural reduction to two dimensional representations, existing optimization for time-critical performance, and the availability of fast implementations for many well-established algorithms. Furthermore, we can carry on perception-based image analysis approaches and mimic human opinion to improve and guide the visual exploration process.

1.2 Overview and Contributions

Parts of this thesis where published in a number of conference proceedings [AEL+09, TAE+09, AEL+10, AEM11, EAM11] and journal articles [ALM11, TAE+11], including VAST and InfoVis conference proceedings. These publications are the basis of this thesis and are integrated under the context of visual analysis of high-dimensional spaces. The contributions are organized into four main parts:

- Chapter 2 presents image-based quality metrics that can be used to select the best projections of multidimensional data sets. The metrics were developed for different visualization methods, including scatterplots, radviz, pixel based visualizations and parallel coordinates. They can effectively support the search for information-bearing views of a data set according to pre-defined user tasks, e.g.

to find correlations between the dimensions or projections of the data that show well-separated clusters.

- Chapter 3 introduces a perception-based quality metric that makes use of the user's opinion.With this contribution, the set of metrics from Chapter 2 is complemented with a metric that mimics the user's perception and which can be trained for different visualization methods and user tasks. Results for scatterplots and for the exploration tasks of finding correlations and clusters are showed.

- Chapter 4 presents novel visualization methods and shows how quality measures can be used to improve the use of visualization matrices. Specifically, a class-based scatterplot matrix, a parallel coordinate matrix and a content aware sorting algorithm for ordinary visualization matrices are introduced. Furthermore, the chapter is concluded with a content-aware color mapping algorithm that can be used to enhance the visual separability of the represented data values.

- Chapter 5 introduces a visual framework for the synthetic generation of high-dimensional data sets. The primary goal of this framework is to ease the creation of test cases for automated visual exploration algorithms and helps to further analyze their behavior and robustness. It can be used to create synthetic data sets simulating important trends, such as multidimensional clusters, correlations and outliers, supporting the creation of complex non-orthogonal trends and classified data sets.

The last chapter concludes this thesis with a discussion about the achieved results and possible future work.

Chapter 2

Visual Quality Metrics

2.1 Motivation

Although many visualization methods support the exploration of high-dimensional data sets, the visual analysis of such data is still a challenging task. The visualization and analysis of multivariate data sets typically involves mapping the data to lower-dimensional embeddings, which is the case for scatterplots or pixel-oriented methods, or determining a placement of the dimensions in multivariate visualizations, as in parallel coordinates [Ins85] or radviz [HGM+97]. The challenge for the analyst is then to find an insightful mapping. However, depending on the number of dimensions and the chosen visualization technique, there may be a very large number of possible projections to be analyzed. To visually explore large data sources, it is essential to support the analyst with tools that help her/him in finding insightful mappings. This can be accomplished by providing an automated analysis of the possible projections of the data. Classical data exploration paradigms require the user to find interesting phenomena in the data interactively by starting with an initial visual representation, as proposed by Shneiderman [Shn96]: "Overview first, zoom/filter, detail on demand". In large-scale multivariate data sets, however, sole interactive exploration becomes ineffective or even infeasible since the number of possible representations grows rapidly with the number of dimensions. Alternatives are needed that help the user to automatically find

effective and expressive visualizations [KMSZ06]. This chapter describes our automated approaches that can be used to support the exploration process. Our motivation to develop quality measures for different visualization types is that visual analysis is usually performed using different visualization methods simultaneously, and that dimensions selected by a quality measure for a specific visualization method do not necessarily produce good projections for other visualization methods.

We present quality metrics for typical analysis tasks based on four different visualization methods: scatterplots, parallel coordinates, radviz and pixel-based visualizations. The basic idea behind the metrics is to automatically identify potentially relevant visual structures from a given set of candidate visualizations of the data. These structures can either be used to determine the relevance of each visualization with respect to a common predefined analysis task or to find the best placement of the dimensions in multivariate visualizations. The user can then use those visualizations with the highest relevance as the starting point of her/his interactive analysis.

2.2 Related Work

In the last years several approaches for selecting good views of high-dimensional projections and embeddings have been proposed. One of the first was *Projection Pursuit* [FT74, Hub85, Fri87]. Its main idea is to search for low-dimensional (one or two-dimensional) projections that expose interesting structures of the high-dimensional data set, rejecting any irrelevant (noisy or information-poor) dimensions. Most solutions from these automated projection pursuit algorithms, however, are not easy to interpret since the presented axes are often a linear combination of existing dimensions of the data set. To exhaustively analyze such a data set using low-dimensional projections, Asimov presented the *Grand Tour* [Asi85] that supplies the user with a complete overview of the data by generating sequences of orthogonal, two-dimensional projections. While navigating the data set, the user may change the viewing direction, creating a movie-like presentation of the whole original space. The problem with this

approach is that an extensive exploration of a high-dimensional data set is cumbersome and time-consuming. A combination of both approaches, Projection Pursuit and the Grand Tour, is proposed in [CBCH95] as a visual exploration system. Since then, different Projection Pursuit indices have been proposed [FFT75, Hub85], but many of these are limited to scatterplot-like projections, and only a few of these techniques consider possible class information of the data.

As an alternative to Projection Pursuit, the *Scagnostics* method [TT85] was proposed to analyze high-dimensional data sets. Wilkinson presented more detailed graph-theoretic measures [WAG05] for computing the Scagnostics indices to detect anomalies in density, shape and trend. These indices could also be used as a ranking for scatterplot visualizations, depending on the analysis task. In this chapter, we present an image-based metric for non-classified scatterplots in order to quantify the structures and correlations between the respective dimensions. This metric could be used as an additional index in a Scagnostics matrix. Similar to Scagnostics, the Rank-by-Feature framework [SS05] presents new indices and an approach where such indices may be used as ranking criteria to sort the one- and two-dimensional projections of a data set.

Koren and Carmel [KC03] propose a method to create useful projections from high-dimensional data sets using linear transformations. Their method also performs class decomposition of the data, resulting in projections with a clearer separation between classes. For our metrics, we chose not to use projection methods that are a combination of the dimensions since users can hardly understand what such combined projection axes represent. Also considering classified data sets, parallel to our work in [TAE+09], Sips *et al.* [SNLH09] developed a class consistency visualization algorithm. Similar to ours, the class consistency method proposes measures to rank lower-dimensional representations. It filters the best scatterplots based on their ranking values and presents them in an ordinary scatterplot matrix. In addition to the measure for non-classified scatterplots, we also propose two measures for classified scatterplots as an alternative to [KC03] and [SNLH09]. Our measures first select the best projections of the data set, compared to embeddings generated by linear combination of the original vari-

ables. This has the advantage that the orthogonal projection axes can be more easily interpreted by the user.

Another important visualization method for multivariate data sets is *parallel coordinates*. Parallel coordinates were first introduced by Inselberg [Ins85] and are used in several tools, e.g. XmdvTool [War94] and VIS-STAMP [GCML06], for visualizing multivariate data. When working with parallel coordinates it is important to decide the order of the dimensions that are to be presented to the user. Aiming at dimension reordering, Ankerst et al. [ABK98] present a method based on similarity clustering of dimensions, placing similar dimensions close to each other. Yang et al. [YWRH03] developed a method to generate insightful projections also based on similarity between the dimensions. Similar dimensions are clustered and used to create a lower-dimensional projection of the data.

In [Guo03] Guo integrates visual and computational metrics for picking and ordering dimensions in parallel coordinates. He describes a human-centered exploration environment which incorporates a combination of computation and visualization methods to explore high-dimensional data and find patterns in these spaces. The main difference between this approach and ours is that Guo searches for locally defined patterns in subspaces while our work concentrates on finding global patterns in a 2-dimensional projection of the data set. As an alternative to the existing methods for dimension reordering of parallel coordinates, we propose a method based on structure that can be visualized on low-dimensional embeddings of the data set. Specially, we create a ranking of all possible two-dimensional parallel coordinates using quality metrics and decide the final order of the dimensions based on it. Three different quality metrics for parallel coordinates are presented (in this chapter), for class and non-class-based visualizations.

Radviz is a radial visualization method. It is similar to parallel coordinates in the sense that it allows to visualize all dimensions of the data set at once. It was first proposed in [HGM$^+$97] to aid in the classification of DNA sequences. Later on, the radviz was extensively used to search for trends, especially clusters, in multidimensional data

sets [HGP99, SGM08, Nv09b, Nv09a]. In this chapter, we investigate the use of quality metrics to define an effective placement of the dimensions for a radviz. Earlier, the dimensions were plotted either in the original order of the data set or using a *Class Discrimination Layout Algorithm* [SGM08]. The *Class Discrimination Layout Algorithm* produces feasible results when applied to flattened data sets [GHLP01], i.e. when the dimensionality is artificially expanded by splitting categorical dimensions into two or more additional dimensions; where a new dimension for each value that the categorical dimension can take is created. Our method is better suited for non-categorical data and can be used for specific user tasks like cluster search (Section 2.6.3).

The last contribution in this chapter is a quality measure to appraise the information content of projections in Pixel-Oriented Displays. Pixel-oriented visualization methods are very popular because they support the visualization of very large data sets. An overview of pixel-oriented visualization techniques is presented in [Kei00]. A first trial on quality metrics in such displays was proposed in [SSK06] where an algorithm to measure the randomness of pixel visualizations was defined based on the entropy of the images. We propose a quality measure to assess the information content of pixel visualizations. Our method works on jigsaw maps [Wat05] and is able to successfully rank the displays according to their overall information content.

2.3 Overview

Some applications involve labeled or classified data. In many data sets a classification of the samples in well know classes exists and we have to take this property into account when designing our ranking functions. When dealing with unclassified data, we search for patterns such as clusters or correlations between the data points. These might reveal important characteristics that can be of interest to the user. Classified data is either given by a known dimension in the data set, or can be classified with clustering algorithms such as k-means [Llo82]. Taking the classification of the data into account may be essential in the exploration process, e.g. when the analyst searches for a projection of the data where the known classes can be individually visualized. In order

to see the structure of classified data, it is necessary for the visualizations to separate the clusters or at least to have minimal overlap. The greater the number of classes, the more difficult the separation.

This chapter describes quality metrics that deal with visualizations of classified and unclassified data. An overview of our approach is presented in Figure 2.1. We start from a given multivariate data set and create low-dimensional embeddings (visualizations). According to the task, there are different visualization methods and different ranking functions that can be applied to these visualizations. The functions are supposed to measure the quality of the views and provide the user with a set of useful visualizations. A list of the presented quality metrics is shown in Table 2.1. For scatterplots on unclassified data, we developed the *Rotating Variance Measure* [TAE+11] which favors *xy*-plots with a high correlation between the two dimensions. For classified data, we propose metrics that take the class information into account for computing the ranking value of the images. For scatterplot visualizations of classified data we developed two methods, a *Class Density Measure* [TAE+11], and a *Class Separating Measure* [TAE+11]. These have the goal to find the best scatterplots by showing the separation between the classes. For parallel coordinates on unclassified data, we propose a *Hough Space Measure* [TAE+11] which searches for interesting patterns such as clustered lines in the views. For parallel coordinates in combination with classified data, we propose two metrics: The *Overlap Measure* [TAE+11] that focuses on finding views with as little overlap as possible between the classes so that the classes separate well, and the *Similarity Measure* [TAE+11] which looks for correlations between the lines. For radviz [AEL+10], a dimension reordering algorithm is proposed to improve the visualization. This reordering algorithm can be used for classified and unclassified data, employing similar metrics to the scatterplot method. Finally, a quality metric for unclassified data in Pixel-Oriented Displays is presented, the *Noise Dissimilarity Measure* [AEL+10]. We tested our *Noise Dissimilarity Measure* with two arrangements for pixel-based visualizations: a jigsaw map and spiral arrangement. The metrics are computed directly on the visualization images.

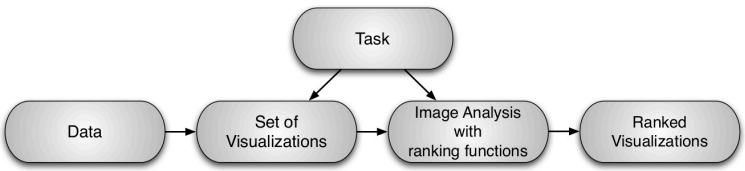


Figure 2.1: Working steps to get a ranked set of good visualizations of high-dimensional data.

2.4 Quality Metrics for Scatterplots

Scatterplots are one of the oldest and most widely used visualization methods [Cha83]. We can define them as graphs where the values of two variables for a sample in a data set are used to plot a point in 2-dimensional space, resulting in a scattering of points. It consists of two axes, the X axis (horizontal axis) and the Y axis (vertical axis), and a series of points that represents the samples of the data set. The position $\mathbf{x} = (x, y)$ of a point p is determined by its X and Y values. Scatterplots are very useful for visually determining the correlation between two variables. When used to explore multivariate data sets, scatterplots are usually visualized in matrix form, also called scatterplot matrix (SPLOM). A SPLOM is a symmetric matrix of adjacent scatterplots that allows the user to analyze the diverse dimensions at once. Figure 2.2 shows an example of SPLOM for a data set of cars [War94]. If there are n variables, the SPLOM has dimension $n \times n$, and the element at the i-th row and j-th column is a scatterplot of the i-th and j-th dimension.

Scatterplots are a very commonly used visualization technique to deal with multivariate data sets to reveal relationships or associations between two dimensions. This low-dimensional embedding of the high-dimensional data in a 2D view can be interpreted easily, especially in the most common case of orthogonal linear projections. Since there are $\frac{n^2-n}{2}$ possible scatterplots for a n-dimensional data set, an automatic analysis technique to preselect the important dimensions is useful or even necessary.

For unclassified data in Section 2.4.1, we propose a quality metric to assess correlation between the dimensions of the scatterplot. For classified data in Section 2.4.2,

Table 2.1: Overview and classification of the presented quality metrics

		Unclassified Data	Classified Data
Scatterplots		Rotating Variance Measure	Class Density Measure Class Separating Measure
Parallel Coordinates		Hough Space Measure	Similarity Measure Overlap Measure
Radviz		Cluster Density Measure	(Same as scatterplots)
Pixel-Based		Noise Dissimilarity Measure	(Not Applied)

our scatterplot quality metrics aim at assessing the density as well as the separateness of classes in the distribution of the data. In the case of unclassified, but well separable data, class labels can be automatically assigned using clustering algorithms [Llo82, Mac67, NJW01].

2.4.1 Scatterplot Metrics for Unclassified Data

In this section we present a quality metric to rank scatterplots of multivariate data sets without class information. This can be used to determine the best views of high-dimensional structures considering correlations between dimensions of the data set.

Rotating Variance Measure High correlations are represented as long, skinny structures in the visualization. Due to outliers even almost perfect correlations can lead to skewed distributions in the plot, and attention needs to be paid to this fact. The *Rotating Variance Measure* (RVM) aims at finding linear and nonlinear correlations between the pairwise dimensions of a given data set.

First, we transform the discrete scatterplot visualization into a continuous density field. For each pixel $\mathbf{p}$ and its position $\mathbf{x} = (x, y)$ the distance to its k-th nearest sample points N_p in the visualization is computed. To obtain an estimate of the local density

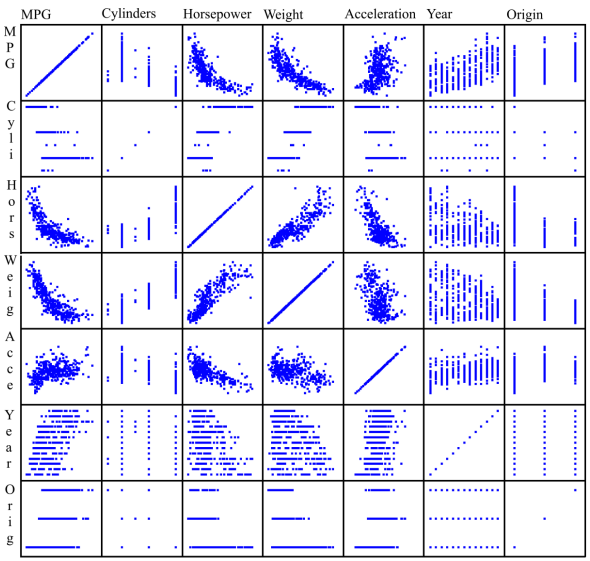


Figure 2.2: Scatterplot matrix example for a data set of cars from [War94]. The element at the i-th row and j-th column is a scatterplot of the i-th and j-th variable

ρ at a pixel $\mathbf{p}$, we define $\rho = 1/r$, where r is the radius of the enclosing sphere of the k-nearest neighbors of $\mathbf{p}$ given by

$$r = max_{i \in N_p} ||\mathbf{x} - \mathbf{x}^i||, \tag{2.1}$$

where $r \in [1, \sqrt{w^2 + h^2}]$ for a scatterplot with at least $k+1$ distinct points and w and h are the width and height of the scatterplot image. Low values for r (densely populated regions in the scatterplot) leads to a small enclosing sphere of the k-nearest neighbors of $\mathbf{p}$ and consequently to a high value for the local density ρ. Choosing the k-th nearest neighbor instead of the nearest neighbor eliminates the influence of outliers and avoids the assumption of a high local density value ρ to these points, e.g. when two points are close to each other but distant from the others in the scatterplot. k is chosen to be between 2 and $n-1$, so that the minimum value of r is mapped to 1. We use $k = 4$ throughout this work. Other density estimations could, of course, be used as well. Vi-

sualizations containing high correlations should generally have corresponding density fields with a small band of larger values, while views with lower correlation should have a density field consisting of many local maxima distributed in the image. We can estimate this amount of distribution for every pixel by computing the normalized mass distribution by taking s samples along different lines l_θ centered at the corresponding pixel positions $\mathbf{x}_{l_\theta}$ and with length equal to the image width, see Figure 2.3. For these sampled lines, we compute the normalized mass distribution for each pixel position $\mathbf{x}^i$ as:

$$v_\theta^i = \frac{\sum_{j=1}^{s} \mathbf{p}_{l_\theta}^{s_j} ||\mathbf{x}^i - \mathbf{x}^{s_j}||}{\sum_{j=1}^{s} \mathbf{p}_{l_\theta}^{s_j}} \tag{2.2}$$

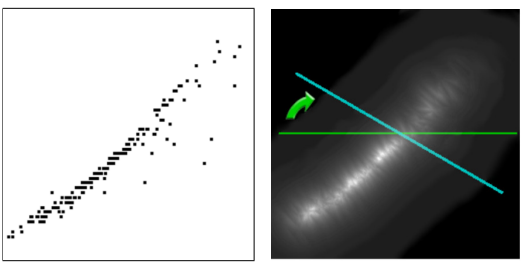
$$v^i = \min_{\theta \in [0,2\pi]} v_\theta^i, \tag{2.3}$$

where $\mathbf{p}_{l_\theta}^{s_j}$ is the density value of the j-th sample along line l_θ and $\mathbf{x}^{s_j}$ is its corresponding position in the image. The distribution value v_θ^i will be very small for pixels positioned at a maximum of a density image if the line l_θ is orthogonal to the main direction of the correlation. Note that such a line can be found even for non-linear correlations. In the opposite case, for positions of the image with low density values, the v^i will present high values. This also implies that pixels in density images conveying low correlation will have large v values.

For each column in this new image composed by v_i values we compute the minimum value and sum up the result. The final RVM value is therefore defined as:

$$RVM = \frac{1}{\sum_x \min_y v(x,y)}, \tag{2.4}$$

where $v(x,y)$ is the mass distribution value at pixel position (x,y). Originally, the $RVM \in (0,\infty)$, but to allow a better comparison between the metrics we normalize it to $[0,1]$. For each data set we normalize the values of the RVM for all scatterplots such that the scatterplot with highest value receives $RVM = 1$ and the scatterplot with the lowest value receives the value $RVM = 0$. A high RVM value stands for a high correlation, while a low value represents a low correlation in the scatterplot.



(a) (b)

Figure 2.3: Scatterplot example and its respective density image. For each pixel we compute the mass distribution along different directions and save the smallest value, in this example depicted by the blue line ≈ orthogonal to the main axis of the correlation.

2.4.2 Scatterplot Metrics for Classified Data

Most known techniques, e.g. the Scagnostics indices [WAG05], calculate the quality of a projection without taking the class distribution into account. To determine the value of a scatterplot of classified or labeled data we can inverstigate its class distribution. Good projections should show good class separation, i.e. minimal overlap of classes.

In this section we propose two approaches to rank scatterplots of multivariate classified data sets in order to determine which views of the high-dimensional structures best separate the different classes.

Class Density Measure The *Class Density Measure* (CDM) evaluates orthogonal projections, i.e. scatterplots, according to their separation properties. Therefore, CDM computes a score for each candidate plot that reflects the separation properties of the classes. The candidate plots are then ranked according to their score so that the user can start investigating highly ranked plots in the exploration process.

In case we are given only the visualization without the data, we assume that every color used in the visualization represents one class. The classes are first separated into distinct images, so that each image contains only the information of one class. Next, we compute a continuous representation for each class to measure the overlap between

the classes. For each pixel $\mathbf{p}$ the distance to its k-nearest neighbors N_p of the same class is computed, and the local density is derived as described earlier in Section 2.4.1.

Having these continuous density functions available for each class, we estimate the mutual overlap by computing the sum of the absolute difference between each pair and sum up the result:

$$CDM = \sum_{k=1}^{M-1} \sum_{l=k+1}^{M} \sum_{i=1}^{P} |\mathbf{p}_k^i - \mathbf{p}_l^i| \quad , \tag{2.5}$$

with M being the number of density images, i.e. classes, $\mathbf{p}_k^i$ being the i-th pixel value in the density image computed for class k, and P being the number of pixels. The original range for the CDM is between 0 and P, but we normalize it to $[0,1]$ to allow a better comparison between the metrics. The CDM value is large if the densities of the different classes at each pixel differ strongly, i.e. if one class has a local high density value compared to all others. It follows that the visualization with the smallest overlap of the classes will be given the highest value. This metric can be used not only to assess well-separated clusters but also dense clusters, which eases interpretability of the data in the visualization. Note that non-overlapping classes in scatterplots produce very different density images using our algorithm. Even if the clusters have similar forms, the density images are different if they do not overlap, resulting in a high value for the CDM metric.

Class-Separating Measure The *CDM* measure finds views that simultaneously show little overlap between classes and that depict dense clusters in high dimensional data sets. The CDM measure is computed over density images with a rapid falloff function. The local density ρ was defined as $\rho = 1/r$ (Section 2.4.1). By changing this function, we are able to control the balance between the property of separation and dense clustering. Choosing a slower decay can yield better-separated clusters but with a lower clustering property.

In our experiments we found that using $\rho = r$ instead $\rho = 1/r$, provides a good trade-off between class separability and clustering. In extension to the *CDM* measure,

we therefore propose the *Class-Separating Measure* (CSM). The main difference between the two measures is in the computation of the continuous representation of the scatterplot, henceforth termed distance field for the *CSM* (with $\rho = r$), and density image for the *CDM* (with $\rho = 1/r$).

To compute a distance field, the local distance at a pixel $\mathbf{p}$ is defined as r, where r is the radius of the enclosing sphere of the k-nearest sample points of $\mathbf{p}$, as described earlier in Section 2.4.1. Once we have the distance field of each class, the CSM is computed as the sum of the absolute difference between them (note that for the CDM, the inverse of the distance was used):

$$CSM = \sum_{k=1}^{M-1} \sum_{l=k+1}^{M} \sum_{i=1}^{P} |\mathbf{p}_k^i - \mathbf{p}_l^i| \quad , \tag{2.6}$$

with M being the number of distance field images, i.e. classes, $\mathbf{p}_k^i$ being the i-th pixel value in the distance field computed for class k, and P being the number of pixels. The original range for the CSM is between 0 and $P\sqrt{w^2+h^2}$, but we normalize it to $[0,1]$ to allow a better comparison between the metrics. It follows that the visualization with the largest distances between the classes will be given the highest value. Comparing the CSM and the CDM, the *Class-Separating Measure* has a bias towards large distances between clusters, while the *Class Density Measure* has a bias towards dense clusters. We consider separation and density of the clusters as two different user tasks. Frequently, views with well-separated clusters are not necessarily the ones with dense clusters. If a view presents both properties simultaneously, it is assigned with a higher value by the two metrics, producing a similar rank for both metrics. A comparison between the CSM and CDM for real data is presented in Section 2.4.3.

2.4.3 Experiments

To evaluate our metrics for scatterplots we test them on a variety of different real data sets. We apply our *Class Density Measure (CDM)* and *Class Separating Measure (CSM)* on classified data to find views that separate or show similarities between classes. For unclassified data, we apply our *Rotating Variance Measure (RVM)* in order

to find linear or non-linear correlations in the data sets. For the visual quality metrics for scatterplots we use the following data sets: *Parkinson's Disease* is a data set composed of 195 biomedical voice measures from 31 people, 23 with Parkinson's disease [LMR+07, LMH+09]. Each of the 23 dimensions is a particular voice measure, e.g *Average vocal fundamental frequency (MDVP:Fo(Hz))*, *Maximum vocal fundamental frequency (MDVP:Fhi(Hz))* and nonlinear measures of fundamental frequency variation. The voice recordings from these individuals have been taken with the goal to discriminate healthy people from those with Parkinson's disease. *Olives* is a classified data set with 572 olive oil samples from nine different regions in Italy [ZNLG94]. For each sample, the normalized concentrations of eight fatty acids are given. The large number of classes (regions) poses a challenging task to the algorithms trying to find views in which all classes are well separated. The *Wisconsin Diagnostic Breast Cancer* (WDBC) data set consists of 569 samples with 30 real-valued dimensions each [SWM93]. The data is classified into malign and benign cells. The task is to find the best separating dimensions. *Wine* is a classified data set with 178 instances and 13 attributes describing chemical properties of Italian wines derived from three different cultivars [ACDV94]. Table 2.2 gives an overview of which data sets are used to evaluate the metrics.

Table 2.2: Overview of data sets that are used to evaluate the quality metrics for scatterplots.

	RVM	**CDM**	**CSM**	**SPLOM**
Parkinson	x			
Olives		x		
Breast Cancer	x	x		x
Wine		x	x	

First we evaluate the RVM on the *Parkinson's Disease* data set. The three best and the three worst results are shown in Figure 2.4. High correlations have been found between the dimensions Dim 9 (DFA - Signal fractal scaling exponent) and Dim 12 (PPE

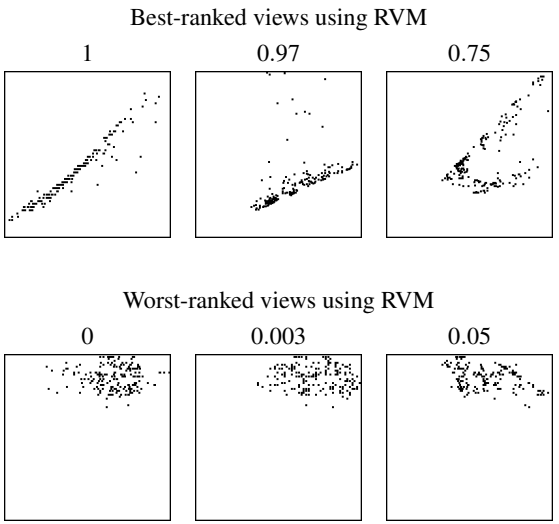


Figure 2.4: Results for the Parkinson's Disease data set using our RVM (Section 2.4.1). Clumpy, low-correlation bearing views are punished (bottom row), while views containing higher correlation between the variables are preferred (top row).

- a nonlinear measure of fundamental frequency variation), Dim 2 (MDVP:Fo(Hz) - Average vocal fundamental frequency) and Dim 3 (MDVP:Fhi(Hz) - Maximum vocal fundamental frequency), as well as Dim 2 (MDVP:Fo(Hz) - Average vocal fundamental frequency) and Dim 4 (MDVP:Flo (Hz) - Minimum vocal fundamental frequency) and got a high value by the metric. Visualizations containing low correlations receive a low value.

In Figure 2.5 the results for the *Olives* data set using our CDM are shown. Even though a view separating all different olive classes does not exist, the CDM reliably proposes three views which separate the data quite well in the dimensions Dim 4 (oleic) and Dim 5 (linoleic), Dim 1 (palmitic) and Dim 5 (linoleic) as well as Dim 1 (palmitic) and Dim 4 (oleic). Palmitic is a saturated fatty acid with 16 carbons(C_16), oleic(C_18) is a monounsaturated fatty acid and linoleic is a polyunsaturated fatty acid [RW95].

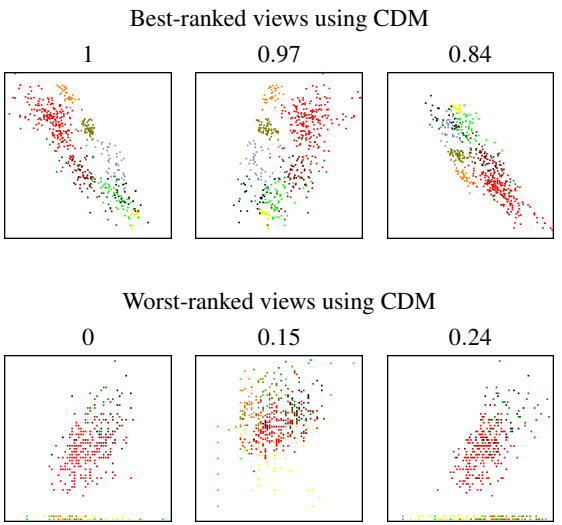


Figure 2.5: Results for the Olives data set using our CDM (Section 2.4.2). Different colors depict different classes (here, growing regions) of the data set. While for this data set, it is impossible to find views completely separating all classes, our CDM still automatically finds views where most of the classes are mutually separated (top row). In the worst-ranked views the classes clearly overlap with each other (bottom row).

Applying the CSM to the Wine data set reveals views that present a good separation between the classes (Figure 2.6). The best ranked plots Dim 7 (Flavanoids) and Dim 13 (Proline), Dim 7 (Flavanoids) and Dim 10 (Color intensity), and Dim 7 (Flavanoids) and Dim 12 (OD280/OD315 of diluted wines) yield a large distance between the centers of the class clusters. The worst ranked views, in contrast, show only cluttered data. The result for the CDM on the Wine data set is depicted in the Figure 2.7. The best ranked plots (Dim 7 (Flavanoids) and Dim 10 (Color intensity), Dim 1 (Alcohol) and Dim 7 (Flavanoids), and Dim 7 (Flavanoids) and Dim 13 (Proline)) present more dense clusters, as expected. The second-best ranked view, Dim 1 (Alcohol) and Dim 7 (Flavanoids) (with CDM = 0.89), would not be considered useful using the CSM alone (CSM = 0.58). Comparing Figure 2.6 and Figure 2.7, we can observe that the

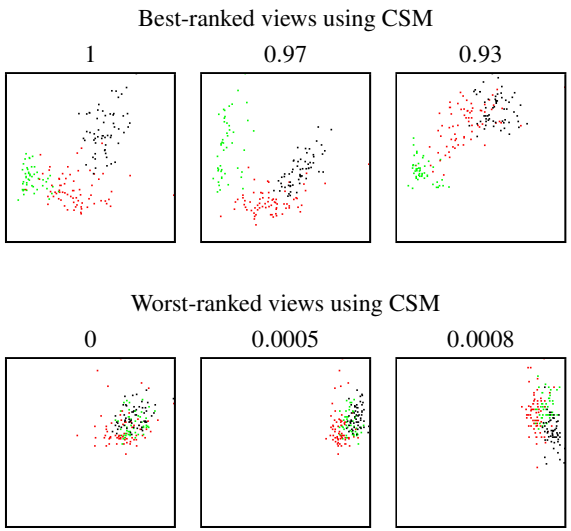


Figure 2.6: Results for the Wine data set using our CSM metric (Section 2.4.2). The best ranked plots yield well separated centers of the class clusters, while in the worst ranked views cluster centers almost coincide.

CSM favors large distances between the clusters, while the CDM assigns high values to views that present dense but separated clusters, even if the distances between them are much smaller.

The analyst has also the possibility to look at all orthogonal views of a data set at once by arranging them in a scatterplot matrix. In our system, the scatterplots are shown in the upper right half of the SPLOM, while the other half is used to display the quality values of each plot. To guide the analysis, the user can fade out lower ranked views, which helps to focus on those with a higher probability of information-bearing content. This is especially helpful if the number of dimensions in the data set is very large, as the number of plots in a SPLOM increases quadratically. Figure 2.8 shows an example. Both SPLOMs show the WDBC data set, but the left one shows the results for the RVM metric while the right one shows the results for the CDM metric. The

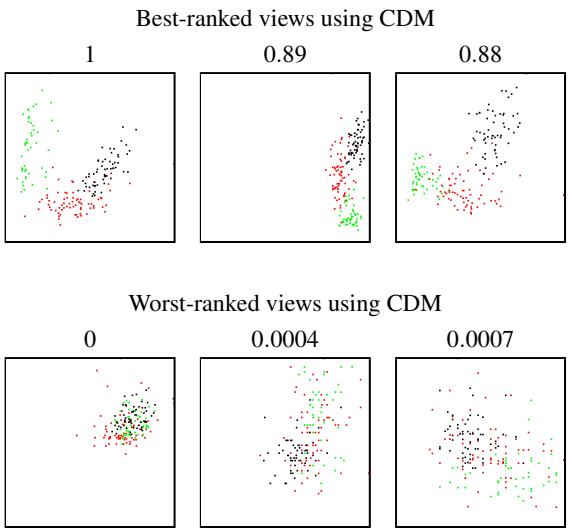


Figure 2.7: Results for the Wine data set using our CDM metric (Section 2.4.2). Note that the second-best ranked view, with Dim 1 (Alcohol) and Dim 7 (Flavanoids) (CDM = 0.89), does not appear in the best ranked plots of the CSM metric (Figure 2.6). The CSM metric value for this plot was only 0.58, because it does not show a large distance between the center of the clusters, in contrast to the best ranked plots in Figure 2.6.

threshold for both SPLOMs was manually set to 0.95 to better visualize the results of the metrics, so all plots with a lower rank have been faded out. As can be seen in the enlarged detail, different views come into focus depending on the chosen metric. While the RVM selects plots with a high degree of correlation, the CDM focuses on separating the designated classes, here the malign and benign cells. Which criterion is more preferable always depends on the user task.

2.5 Quality Metrics for Parallel Coordinates

Parallel Coordinates is another well-known and widely used visualization method for multivariate data sets [Ins85]. It consists of placing the set of axes parallel to each

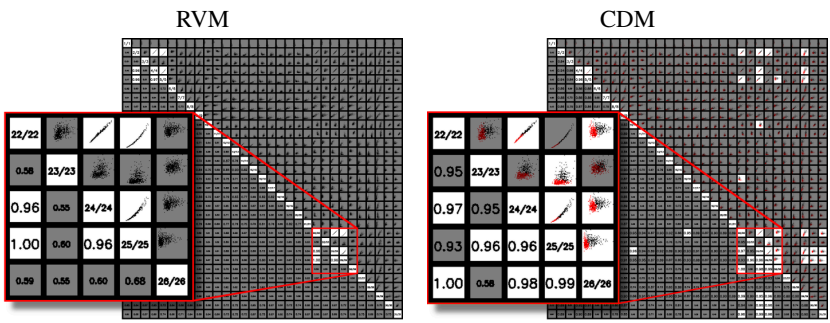


Figure 2.8: Results on the WDBC data set for the RVM (left) and the CDM (right). The threshold for both SPLOMs was manually set to 0.95 to better visualize the results of the metrics. Views with a metric value of less than this threshold have been faded out. This way many irrelevant views can be ignored.

other, where each axis represents one dimension of the data set. Each sample of an N-dimensional data set is represented by a polyline that intersects all N vertical axes (dimensions). The intersection point represents its value in the respective dimension. Figure 2.9 shows an example of a parallel-coordinates plot for a car data set from [War94].

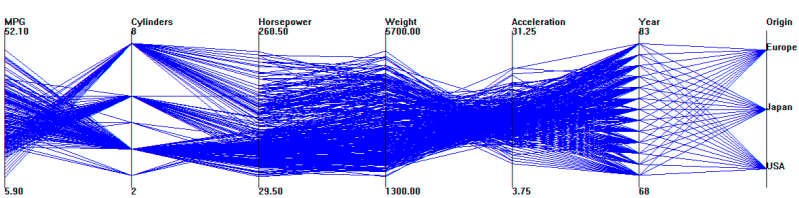


Figure 2.9: Parallel coordinates example for a cars data set from [War94]. Each axis represents one dimension of the data set and each sample of an N-dimensional data set is represented by a polyline that intersects N vertical axes.

One problem of this kind of visualization is the large number of possible permutations of the dimension axes. For an n-dimensional data set $\frac{n+1}{2}$ permutations are

needed to visualize all adjacencies [Weg90], but there are $n!$ possible arrangements. An automated analysis of the visualizations can help to find the best visualizations out of all possible arrangements, however, for high-dimensional data sets, analyzing $n!$ visualizations is practically impossible: for $n = 20$ we have already $2.4 \cdot 10^{18}$ possible mappings. Constrained by this limitation, we attempt to analyze only the pairwise combinations of dimensions which are later assembled to find the best visualizations, reducing the number of arrangements to n^2 visualizations.

When analyzing parallel-coordinate plots, we focus on the detection of plots that show either significant correlation between attribute dimensions, or good clustering properties in certain attribute ranges.

2.5.1 Parallel Coordinate Metrics for Unclassified Data

In this section we present a quality metric to rank two-dimensional parallel coordinates of multivariate data sets without class information which can be used to determine the best views of high-dimensional structures considering clustered lines in the two-dimensional projections of the data set.

Hough Space Measure Our analysis aims at finding clustered lines with similar positions and directions. Our algorithm for detecting these clusters is based on the Hough transform [Hou62].

Straight lines in the image space can be described as $y = ax + b$. The main idea of the Hough transform is to define a straight line according to its parameters, i.e. the slope a and the interception b. Due to a practical difficulty (the slope of vertical lines is infinite) an alternative representation of a line is chosen:

$$\rho = xcos\theta + ysin\theta, \tag{2.7}$$

where ρ is the length of the normal from the origin to the line and θ is the angle between this normal and the x-axis. Using this representation, for each non-background pixel in the visualization we have a distinct sinusoidal curve in the $\theta\rho$-plane, also

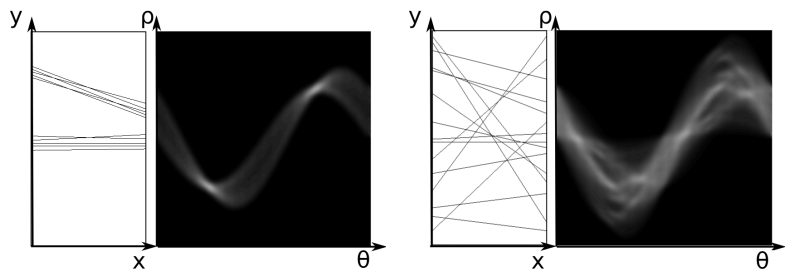


Figure 2.10: Synthetic examples of parallel coordinates and their respective Hough spaces: (a) presents two well-defined line clusters and is more interesting for the cluster identification task than (b), where no line cluster can be identified. Note that the bright areas in the $\theta\rho$-plane represent the clusters of lines with similar θ and ρ.

called Hough or accumulator space. An intersection of many such curves indicates that the corresponding pixels belong to the same line defined by the parameters (θ_i, ρ_i) in the original image space. Figure 2.10 shows the Hough transform of two synthetic examples of parallel coordinates and their respective Hough spaces: Figure 2.10a presents two well-defined line clusters, while in Figure 2.10b, no line cluster can be identified: the bright areas in the $\theta\rho$-plane represent clusters of lines with similar θ and ρ.

To reduce the bias towards long lines, we scale the pairwise visualization images to $n \times n$ resolution, usually 512×512 pixels. The accumulator space is quantized into a $w \times h$ cell grid, where w and h control the similarity sensitivity of the lines. We use 50×50 grid cells in our examples. A lower value for w and h reduces the sensitivity of the algorithm because lines with a slightly different θ and ρ are mapped to the same accumulator cells.

Based on our definition, good visualizations contain a few well-defined clusters, which are represented by accumulator cells with high values. To identify these cells, we compute the median value m as an adaptive threshold that divides the accumulator function $h(\mathbf{x})$ into two identical parts:

$$\begin{aligned} \frac{\sum h(\mathbf{x})}{2} &= \sum g(\mathbf{x}) \quad \text{, where} \\ g(\mathbf{x}) &= \begin{cases} \mathbf{h} & \text{if } \mathbf{h} \leq m; \\ m & \text{else,} \end{cases} \end{aligned} \tag{2.8}$$

with $\mathbf{x}$ being the position of a pixel $\mathbf{h}$ in the accumulator function $h(\mathbf{x})$. Using the median value, only a few clusters are selected in an accumulator space with high contrast between the cells (Figure 2.10a), while in a uniform accumulator space many clusters are selected (Figure 2.10b). This adaptive threshold is not only necessary to select possible line clusters in the accumulator space, but it also reduces the influence of outliers and occlusion between the lines. In the occlusion case, a point that belongs to two or more lines is computed just once in the accumulator space.

The final goodness value for a 2D visualization is computed by the number of accumulator cells n_{cells} that have a higher value than m, normalized by the total number of cells $(w \times h)$ to the interval $[0,1]$:

$$s_{i,j} = 1 - \frac{n_{cells}}{wh}, \tag{2.9}$$

where i, j are the indices of the respective dimensions, and the computed metric $s_{i,j}$ yields higher values for images containing well-defined line clusters (similar lines) and lower values for images containing lines in many different directions and positions.

Having combined the pairwise visualizations to create a complete parallel coordinates plot, we can now compute the overall quality metric by summing up the respective pairwise measurements. This overall quality metric of a parallel visualization containing n dimensions is:

$$HSM = \sum_{a_i \in I} s_{a_i, a_{i+1}}, \tag{2.10}$$

where I is a vector containing any possible combination of the n dimensions indices. In this way, we can measure the quality of any given parallel coordinates plot.

Exhaustively computing all n-dimensional combinations in order to choose the best/worst ones requires very long computation times and becomes unfeasible for large

n. In these cases, in order to search for the best n-dimensional combinations in a feasible time frame, an algorithm to solve the Traveling Salesman Problem is used, e.g. the A*-Search algorithm [HNR68] or others [ABCC07]. Instead of exhaustively combining all possible pairwise visualizations, this kind of algorithm composes only the best overall arrangement of dimensions for the parallel coordinates.

2.5.2 Parallel Coordinates Metrics for Classified Data

While analyzing parallel coordinates visualizations with class information, we consider two main issues. First, in good parallel coordinates visualizations, the lines that belong inside a determined class must be similar (inclination and position similarity). But also, visualizations where the classes can be observed separately and that contain less overlap are also considered to be good. We develop two metrics for classified parallel coordinates that take these matters into account: the *Similarity Measure* that encourages inner-class similarities, and the *Overlap Measure* that analyzes the overlap between classes. Both are based on the metric for unclassified data in parallel coordinates presented in Section 2.5.1.

Similarity Measure The similarity measure is a direct extension of the metric presented in Section 2.5.1. For visualizations containing class information, the different classes are usually represented by different colors. We separate the classes into distinct images that contain only the pixels in the respective class color and compute a quality metric s_k for each class using Equation (2.9). Thereafter, an overall quality value s is computed as the sum of all class quality metrics:

$$SM = \sum_k s_k. \tag{2.11}$$

Using this metric, we encourage visualizations with strong inner-class similarities and slightly penalize overlapping classes. Note that due to class overlap, some classes have many missing pixels which results in a lower s_k value compared to other visualizations with less or no overlap between classes.

Overlap Measure In order to penalize overlap between classes, we analyze the difference between the classes in Hough space (see Section 2.5.1). As in the similarity measure, we separate the classes into different images and compute the Hough transform over each image. Once we have a Hough space h for each class, we compute the quality metric as the sum of the absolute difference between the classes:

$$OM = \sum_{k=1}^{M-1} \sum_{l=k+1}^{M} \sum_{i=1}^{P} |\mathbf{h}_k^i - \mathbf{h}_l^i|. \tag{2.12}$$

Here M is the number of Hough space images, i.e. classes, and P is the number of pixels. This OM value is high if the Hough spaces are disjoint, i.e. if there is no large overlap between the classes. Therefore, the visualization with the smallest overlap between classes receives the highest value.

Another interesting application of this metric is to search for similarities between different classes. In this case, overlap between classes is desired, and the previously computed metric can be inverted to compute suitable quality values:

$$OM\_INV = 1/OM. \tag{2.13}$$

2.5.3 Experiments

To measure the merits of our approaches for parallel coordinates, we test them using two different real and one synthetically generated data set: *Cars* contains 7404 cars listed with 23 different attributes, including type of motor (the class of the data set), manufacturer, type, price, cylinder capacity, power, rpm, torque, vmax, acceleration, fuel consumption, CO_2 emission, weight, length, width, height, wheel base, load capacity, trunk, towing capacity, roof load, tank capacity and taxes. The *Cars* data set was automatically collected from a national second-hand car selling website by the Institute for Information Systems from the Technische Universität Braunschweig (Germany). The *Wisconsin Diagnostic Breast Cancer* (WDBC) data set, that was already used to evaluate the scatterplots in Section 2.4.3, consists of 569 samples with 30 real-valued dimensions each [SWM93]. The data is classified into malign and benign cells.

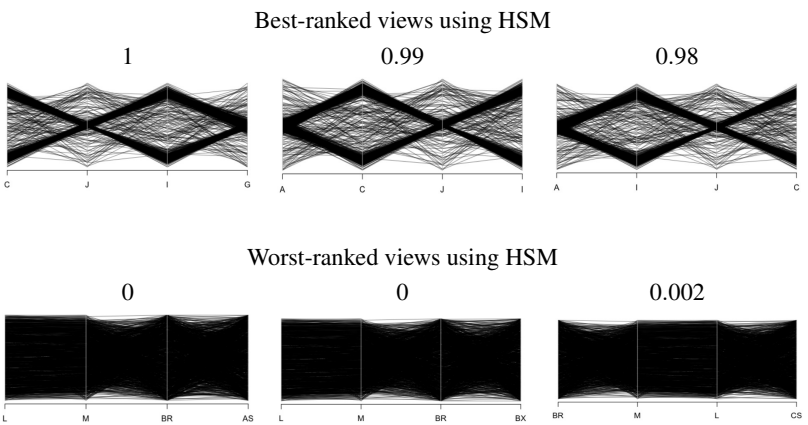


Figure 2.11: Results for the *synthetic* data set [JJ09]. Best and worst-ranked visualizations using our HSM metric for non-classified data (ref. Section 2.5.1). (a) Top row: The three best-ranked visualizations and their respective normalized metrics. Well-defined clusters in the data set are favored. Bottom row: The three worst-ranked visualizations. The large amount of spread exacerbates interpretation. Note that the user task related to this metric is not to find high correlation between dimensions but to detect well-separated clusters.

The task is to find the best separating dimensions. Finally, to compare our metric to the method proposed in [JJ09], we use their synthetic data set that contains 1320 data items and 100 dimensions, of which 14 dimensions contain significant structures.

We estimated the best and worst-ranked visualizations of different data sets. The corresponding visualizations are shown in Figures 2.11, 2.12 and 2.13. For better comparability, the visualizations have been cropped after the display of the 4th dimension. We used a size of 50×50 cells for the Hough accumulator in all experiments. The algorithms are quite robust with respect to the size, and using more cells generally only increases computation time but has little influence on the result.

Recent work presented by Johansson and Johansson [JJ09] introduces a system for dimensionality reduction by combining user-defined quality metrics using weighted functions to preserve as many important structures as possible. The analyzed structures

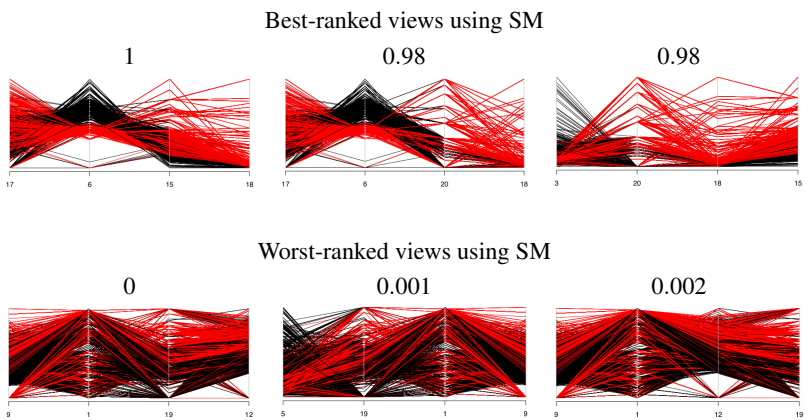


Figure 2.12: Results for the *Cars* data set. Cars using gasoline are shown in black, diesel in red. Best and worst-ranked visualizations using our Hough Similarity Measure (Section 2.5.2) for parallel coordinates. (a) Top row: The three best-ranked visualizations and their respective normalized metrics. Bottom row: The three worst-ranked visualizations.

are clustering properties, outliers, and dimension correlations. We use a synthetic data set presented in their paper to test our *Hough Space Measure*. As aforementioned in section 2.5.1, the HSM algorithm prefers views that show similarity between the lines. We compute our HSM on this synthetical data set and present the result in Figure 2.11. Here we can see the best-ranked plots for clustered data points in the top row and the worst-ranked plots in the bottom. At the top, the clusters of lines are clearly visible in contrast to the bottom where no structures are visible. The five dimensions that are in the best plots are dimensions **A, C, G, I, J**. Four out of five dimensions are also determined by [JJ09] as the best dimensions for clustering. They use user-defined quality metrics for their system. Our resulting dimensions are a subset of their best 9 dimensions. This indicates that our metrics are comparable to user-defined metrics and are able to rank plots in a similar way users would do.

When the *Hough Similarity Measure* is applied to the *Cars* data set, we can see that there seem to be barely any good clusters in the data set (see Figure 2.12). We

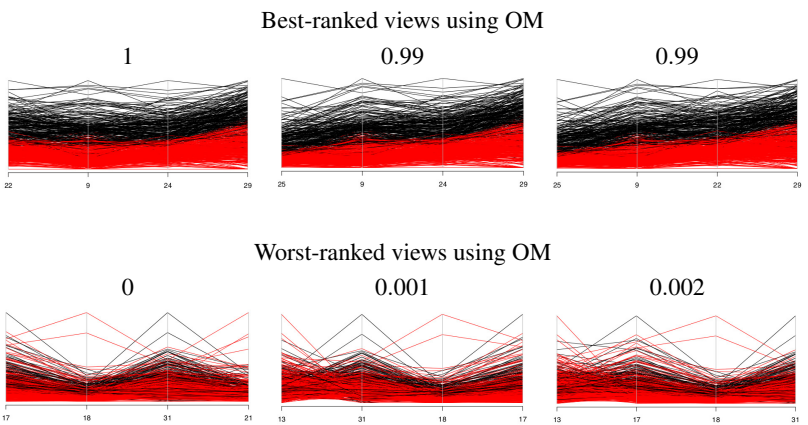


Figure 2.13: Results for the *WDBC* data set. Malign nuclei are colored black while healthy nuclei are red. Best and worst ranked visualizations using our Overlap Measure (Section 2.5.2) for parallel coordinates. (a) Top row: The three best-ranked visualizations. Despite similarity between the lines, visualizations that minimize the overlap between classes are favored, so the difference between malign and benign cells becomes clearer. Bottom row: The three worst ranked visualizations. The overlap of the data complicates analysis, the information is useless for the task of discriminating malign and benign cells.

verify this observation by exhaustively looking at all pairwise projections. The only dimension where the classes can be separated and at least some form of cluster can be reliably found is Dim 6(RPM), in which cars using diesel generally have a lower value compared to benzine (Figure 2.12, top row). The similarity of the majority in Dim 15(Height), Dim 18(Trunk) and Dim 3(Price) can also be detected. Obviously cars using diesel are cheaper. This might be due to the age of the diesel cars, but age was unfortunately not included in the data base. On the other hand, the worst-ranked views using the HSM (Figure 2.12, bottom row) are barely interpretable, at least we were unable to extract any useful information.

In Figure 2.13, the results for our *Overlap Measure* applied to the *WDBC* data set are shown. This result is very promising. In the top row, showing the best plots, the malign and benign are well separated. It seems that the dimensions Dim 22 (fractal

dimension (standard error)), Dim 9 (concavity (mean)), Dim 24 (texture (worst)), Dim 29 (concavity (worst)) and Dim 25 (perimeter (worst)) separate the two classes well.

2.6 Quality Metrics for Radviz

Radviz [HGM+97] is a radial visualization method where the dimensions are represented by points placed equally spaced around a circumference. Each sample $\mathbf{x}_i$ of an n-dimensional data set is represented by a point $\mathbf{p}_i$ in a 2-dimensional plot, as depicted in Figure 2.14. Imagine that each point $\mathbf{p}_i$ is connected by n springs to the n respective dimensions of the data set ,and the spring constant K_i is equal to the j-th coordinate of $\mathbf{x}_i$, namely $x_{i,j}$. The final position of $\mathbf{p}_i$ in the visualization is determined by the point where the sum of all spring forces is zero and can be computed as:

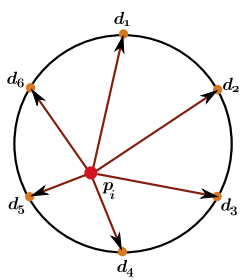


Figure 2.14: Radviz example. The dimensions j are represented by points, placed equally spaced around a circumference and each sample $\mathbf{x}_i$ is plotted at position $\mathbf{p}_i$ according to its coordinate values $x_{i,j}$

$$\mathbf{p}_i = \frac{\sum_{j=1}^{n} \mathbf{d}_j x_{i,j}}{\sum_{j=1}^{n} x_{i,j}} , \tag{2.14}$$

where $\mathbf{d}_j$ is the vector pointing from the center to the position of the respective dimension on the circumference.

An important aspect of the radviz visualization method is that it supports visualizing all dimensions of a data set at once, such that it can be very useful while searching for clusters and outliers in high-dimensional data. Similar to parallel coordinates, a

very important issue in radviz is to decide in which order the dimensions shall be arranged to support a specific user task. Radviz is quite sensitive to the order of the dimensions, e.g. if dimensions with high values for a sample are placed close to each other in one sector on the circumference, this sample point is drawn towards this sector. Similarly, samples with similar coordinate values are plotted close to the center.

We propose to make use of quality metrics to generate a radviz with an appropriate order of dimensions for a specific user task. A quality metric can be successfully applied to a visualization to appraise its information-bearing content, but exhaustively computing all n-dimensional combinations in order to choose the best one requires a prohibitive amount of time, at least for high-dimensional data sets.

The problem at hand is twofold. First, we need a useful quality metric to define whether a specific radviz visualization provides useful information. This is discussed in Section 2.6.1. Second, we need an efficient algorithm to guide the synthesis as it is infeasible to create all possible visualizations. We describe our approach in Section 2.6.2.

2.6.1 Quality Metrics

Diverse quality metrics can be used to quantify the amount of information of a radviz, given a specific user task. Since the samples in a radviz are represented as a scatter of points, most quality metrics for scatterplots may be applied to radviz as well. Our approach is based on the user task of searching for clusters, which is a very common task while visually exploring data sets with the radviz method. For class-based data sets, we make use of the *Class Density Measure* (CDM) described in Section 2.4.2. The CDM favors projections where the defined classes are well separated from each other and penalizes overlapping classes. For the non-class-based data sets, we present a new quality metric to rank visualizations by searching for projections with well-defined clusters. We call this metric *Cluster Density Measure*. Note that for our quality metrics nomenclature, we assume clusters to be groups of data points close together in

the visualization, while classes are defined as groups of data points with a previously know labeling.

Cluster Density Measure The *Cluster Density Measure* (C_lDM) is designed to rank visualizations based on their clustering properties. Besides point-cloud-like visualizations like scatterplots and radviz, it can also be directly applied to dense visualizations like Continuous Scatterplots or Pixel-Oriented Displays. The C_lDM algorithm is directly applied to a visualization image and consists of two main parts: an image clustering algorithm and the metric estimation based on the cluster properties. Figure 2.15 gives an overview of the different steps of the algorithm.

For point-cloud-like visualizations (Figure 2.15a), we first compute a continuous representation of the image, whereupon a density image (Figure 2.15b) is computed based on local neighborhoods in the original image. Working with a density image instead of working with the data points directly has the advantage of not treating outliers as compact clusters. The density at a pixel $\mathbf{p}_i$ is defined as $1/r$, where r is the radius of the enclosing sphere of the k-nearest neighbors of $\mathbf{p}_i$, as described in Section 2.4.1. As these density images are usually still quite noisy, we extract the low-frequency parts in order to create smooth density images (Figure 2.15c). This can be achieved by applying a Gaussian filter with a large standard deviation σ.

Clusters in these density images appear as smooth blobs (Figure 2.15c). The border of a cluster is defined using the second derivative of the smooth density image. They can be conveniently found by convolving the image with a Laplace filter and then searching for zero crossings (Figure 2.15). Finally, the number of clusters is defined based on the distance between the contour points. Two contour points are considered to belong to the same cluster if the distance between them is either smaller than a threshold τ or there exists a path along other contour points where the maximum distance between two adjacent points on the path is always smaller than τ. Otherwise, the contour points belong to different clusters. Based on our experiments, we set $\tau = \sqrt{P}/5$ where P is the number of pixels in the density image. After labeling the contours,

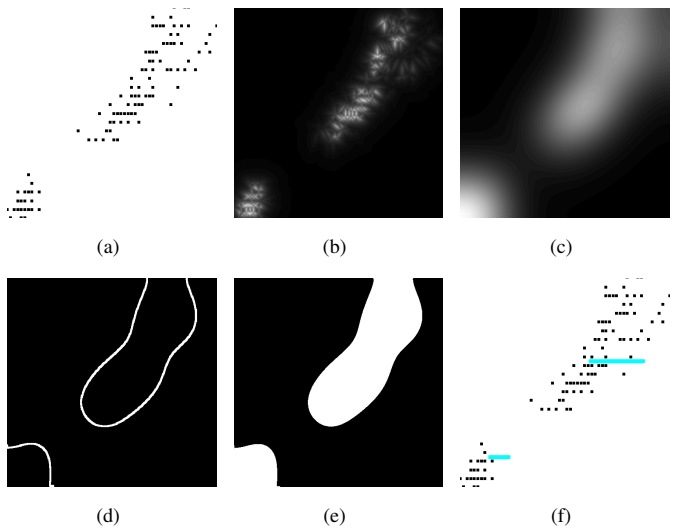


Figure 2.15: C_lDM algorithm example. (a) Original data plot, (b) density image computed based on local neighborhoods in the original image, (c) smoothed density image, (d) cluster contours obtained by zero crossing detection in the Laplace image of (c), (e) detected cluster regions and (f) original image with the average radius per cluster overlaid in the image.

the center $\mathbf{c}_k$ and average radius r_k per cluster are computed.The final metric is then defined as:

$$C_lDM = \frac{1}{K}\sum_{k=1}^{K}\sum_{l=k+1}^{K}\frac{d_{k,l}^2}{r_k r_l}, \tag{2.15}$$

where K is the number of detected clusters and $d_{k,l}$ is the Euclidian distance between the cluster centers $\mathbf{c}_k$ and $\mathbf{c}_l$. Accordingly, the C_lDM assigns high values to views that present well-defined clusters with small intra-cluster distances and large inter-cluster distances.

2.6.2 Radviz Sorting

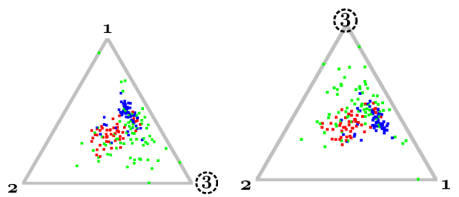
Given the CDM and C_lDM, we are able to quantify the quality of a radviz plot. Now we need an efficient way to find a good radviz plot without creating each possible visualization. We propose a greedy incremental algorithm to successively add dimensions to a radviz plot to define a suitable order. This greedy approach provides a tradeoff between finding the optimal solution, which can be found by exhaustively searching all possible visualization arrangements, and completing all computations in feasible time.

We start by creating a radviz with only two dimensions. The first two dimensions added to the radviz can be the first two in the data set or the best two dimensions determined by some of the quality metrics. We then add another dimension and create all possible 3D radviz plots (at this stage only two positions are possible, see Figure 2.16a). According to the quality metric used, the best sequence of dimensions is selected for further processing. This intermediate radviz is then successively augmented with all other dimensions by searching for and then keeping the best sequence with every dimension added. The final sequence defines a good placement of dimensions according to the chosen metric and user task.

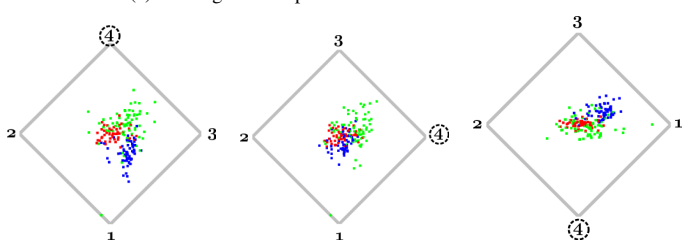
Figure 2.16 shows an example of dimension placement for the first four dimensions of the *Wine* data set [ACDV94]. The algorithm is initialized with the 1^{st} and 2^{nd} dimension and the best placement for the 3^{rd} is computed (Figure 2.16a). It is worth noting that a radviz with three dimensions is not sensitive to their placement, the possible arrangements present only rotated and/or mirrored variations of the same structure. We then create all possible radviz visualizations by adding the fourth dimension. The best ordering is kept. The overall complexity of the proposed algorithm is $O(n^2)$, which is comparably low to an $O(n!)$ exhaustive search.

2.6.3 Experiments

We test our radviz dimension placement algorithm on a variety of data sets. For classified data sets, we apply the *Class Density Measure* (CDM) defined in Section 2.4.2



(a) Finding the best placement for the 3rd dimension.



(b) Finding the best placement for the 4th dimension.

Figure 2.16: Dimension placement example for the first four dimensions of the *Wine* data set. (a) The algorithm is initialized with the 1st and 2nd dimension, and the best placement for the 3rd is computed. (b) Other dimensions are successively added and the best arrangement is kept in each step.

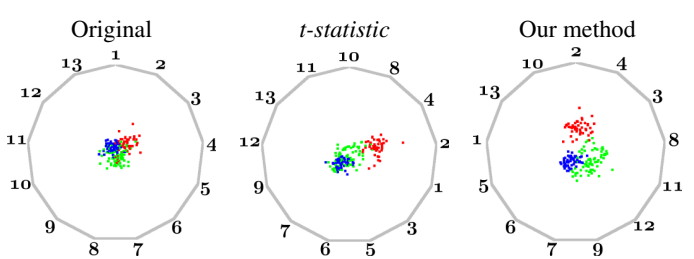


Figure 2.17: Original radviz, t-statistic and our results, respectively, for the *Wine* data set using the CDM metric. The different colors depict the different classes (cultivars) of the data set.

and for unclassified data sets the *Cluster Density Measure* (C_lDM) defined in Section 2.6.1. First, we show our results for the *Wine* data set [ACDV94], a class-based data

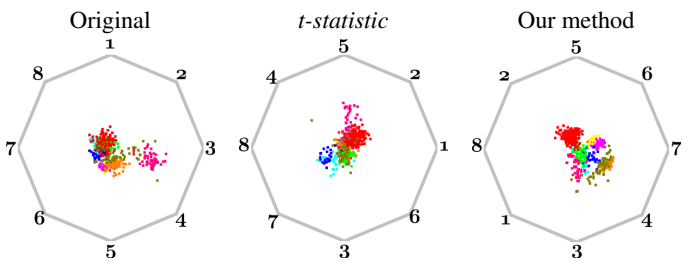


Figure 2.18: Original radviz, t-statistic and our results, respectively, for the *Olives* data set using the CDM metric. The different colors depict the different classes (regions) of the data set.

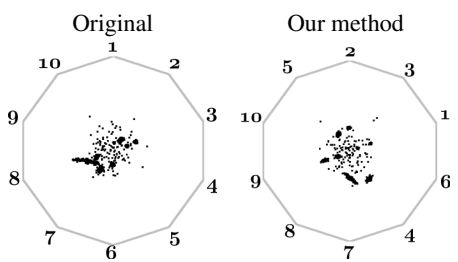


Figure 2.19: Original radviz and our results, respectively, for a synthetic data set with 10 dimensions and 8 clusters using the C$_l$DM metric.

set with 178 records and 13 dimensions that describes chemical properties of Italian wines from different cultivars. The first plot in Figure 2.17 is the original radviz, without dimension replacement. In the second plot, the dimensions are reordered using the t-statistic algorithm proposed in [SGM08]. In the third plot the results of our placement algorithm are shown. The t-statistic is used to group similar dimensions of a data set and requires a classification of the records in some manner. Note that the t-statistic method presents a better plot than the original one with respect to cluster separation task, but the best plot for the same task was achieved using our method. A second example for another class-based data set is shown in Figure 2.18. *Olives* [ZNLG94] is a classified data set with 572 olive oil records from nine different regions in Italy,

which represent the different classes of the data set. For each sample, the normalized concentrations of eight fatty acids are given as attributes.

For unclassified data, we show our results for a synthetic data set with ten dimensions, Figure 2.19. As the t-statistic method requires a classification of the data set, we compare our results for unclassified data only with the original radviz. The left plot presented in Figure 2.19 is the original radviz without any dimension reordering, and the right one is the radviz generated by our dimension placement algorithm. Note that the resulting plot using our C_lDM method presents well-separated clusters unlike the original plot.

2.7 Quality Metrics for Jigsaw Maps

Standard projection techniques reduce the number of dimensions from n to two plus color information for visualization of the data on the screen. In some cases, it turns out to be beneficial to go the other way around and represent a one-dimensional function as a two-dimensional plot in order to preserve some of the characteristics inherent in the data, e.g., natural order and locality. Examples of such data could be the household income of a certain region, or weather data. This is the idea behind Wattenberg's jigsaw maps [Wat05]. Jigsaw maps project the one-dimensional data, or each dimension of multivariate data, into the two-dimensional plane, using a space filling curve in such a way that properties like locality and clusters are preserved. An example can be seen in Figure 2.20. If the data set consists of more than one dimension, one jigsaw map for each dimension is created.

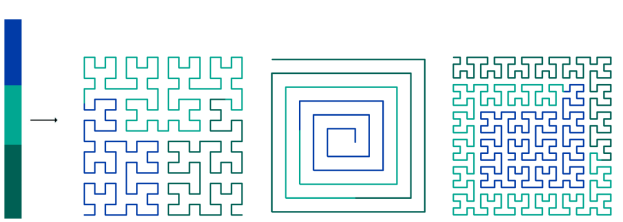
An important step to create a jigsaw map from a set of one-dimensional data points $\mathbf{X}_j$ is to first normalize the values according to the desired output image size s^2: For this purpose a function $g(\mathbf{X}_j)$ needs to be defined to map $\mathbf{X}_j$ onto sequences of subsets of $\{1,2,...,s^2\}$

$$g(\mathbf{X}_j) = (\{1,2,...,m_1\},\{m_1+1,...,m_2\},...,\{m_{k-1}+1,...,m_k\})$$

where $m_i = x_{1,j} + x_{2,j} + ... + x_{i,j}$ and $m_K = s^2$, where k is the number of sequences, j is the dimension of the data set, and s^2 is the number of pixels in the output visualization. Width and height are considered to be of equal size and are usually a power of two. The layout function J for the jigsaw maps can be defined using g and another function H: a screen-filling curve satisfying c-locality. C-locality is preserved if the diameter of a region r_i corresponding to a data point $x_{i,j}$ is bounded by a small constant c. This keeps regions relatively compact in the output. Then

$$J(\mathbf{X}_j) = H(g(\mathbf{X}_j)) \ ,$$

i.e. each data value is given a set of connected positions along the screen filling curve, and a color. A common choice for the color mapping is to assign the data values to a color gradient to ease interpreting the underlying data. We use linear mapping for the experiments in this section. However, other sophisticated mappings as we present in Chapter 4 may be used without additional effort together with our quality metrics. Three examples for space-filling curves and their colorization g is given in Figure 2.20: (b) depicts an H-curve that is used to create a jigsaw map $J(\mathbf{X}_j)$, (c) shows a spiral space-filling curve $S(\mathbf{X}_j)$, and (d) shows a hybrid space-filling curve $S_H(\mathbf{X}_j)$ containing elements of (b) and (c).



(a) g (b) $J(\mathbf{X}_j)$ (c) $S(\mathbf{X}_j)$ (d) $S_H(\mathbf{X}_j)$

Figure 2.20: Three examples for space-filling curves and their colorization g (a): (b) depicts an H-curve that is used to create a jigsaw map $J(\mathbf{X}_j)$, (c) shows a spiral space-filling curve $S(\mathbf{X}_j)$, and (d) shows a hybrid space-filling curve $S_H(\mathbf{X}_j)$ containing elements of (b) and (c).

2.7.1 Noise Dissimilarity Measure

The task at hand is to find interesting structures in a jigsaw map. Clusters are the most well-known structures in such visualizations due to the possibility of conserving locality information of the data. However, it can be hard to find clusters in jigsaw maps as these structures usually do not have any specific size or layout which makes it difficult to describe them in a mathematical sense. Schneidewind *et al.* [SSK06] use the entropy or standard deviation of the color values in different grid cells to derive a quality metric. The algorithm considers regions as interesting if the entropy or standard deviation is larger than a certain threshold τ_{min} but smaller than another threshold τ_{max}. There are two drawbacks to this method. First, the thresholds need to be set by hand. Without any knowledge of the underlying data this can be a time-consuming task. Second, entropy as well as standard deviation do not pay any attention to the spatial arrangement of the data. To do so, the user is urged to provide a weighting function for the hierarchical analysis [SSK06]. Again this can be difficult if the structures searched for are unknown.

Instead of searching for interesting structures directly, we propose to quantify the *dissimilarity* to a noise function $\aleph(\mathbf{X}_j)$. We impose $\aleph(\mathbf{X}_j)$ to have the same color probabilities as $J(\mathbf{X}_j)$, i.e. the histograms of both images are equal. In practice, we can easily achieve this by randomly permutating the pixel coordinates of $J(\mathbf{X}_j)$. An example of such a jigsaw map $J(\mathbf{X}_j)$ and its corresponding noise function $\aleph(\mathbf{X}_j)$ is given in Figure 2.21 in the left and middle images.

Next, we assume that jigsaw maps $J(\mathbf{X}_j)$ as well as the noise function $\aleph(\mathbf{X}_j)$ can be modeled by a Markov Random Field that models the images as a realization of a local and stationary random process. Each pixel is characterized by a small set of spatially neighboring pixels, and this characterization is the same for all pixels. This model has been successfully used for exemplar-based texture synthesis [WLKT09], which is related to our problem. In exemplar-based texture synthesis, one starts with a noise function and tries to explain this noise function with a given input image. Now,

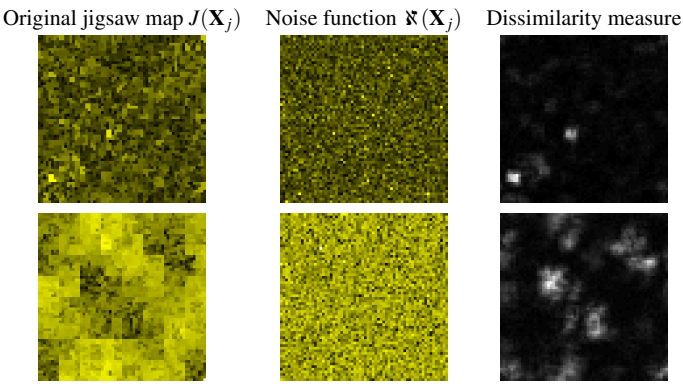


Figure 2.21: Visualization of noise dissimilarity, from left to right: Original jigsaw map $J(\mathbf{X}_j)$, corresponding noise function $\aleph(\mathbf{X}_j)$, and our *Noise Dissimilarity Measure* computed for each pixel (values are linearly scaled for better readability). Lighter values correspond to a stronger dissimilarity and therefore more interesting structures. The top row shows an outlier example in the *Ozone* data set [ZF08]. Our algorithm highlights these outliers as interesting regions (right). In the bottom row, a visualization with more abstract patterns is shown. Note that most of these structures are also captured well by our approach.

we do the same but with exchanged images. We in essence, try to explain a given visualization $J(\mathbf{X}_j)$ with potential information by a given noise function $\aleph(\mathbf{X}_j)$.

Due to the assumed locality and stationary characteristics of our images, we can base our quality metric on the similarity and respectively on the dissimilarity, between local neighborhoods of each pixel in $J(\mathbf{X}_j)$ and each pixel in $\aleph(\mathbf{X}_j)$. We call this the *Noise Dissimilarity Measure* (NDM). We denote a neighborhood of pixels with radius r around a pixel i with $J_{NH(i)}(\mathbf{X}_j)$, and $\aleph_{NH(i)}(\mathbf{X}_j)$, respectively. To quantify noise

dissimilarity, we compute:

$$
\begin{aligned}
NDM(J(\mathbf{X}_j), \aleph(\mathbf{X}_j)) &= \frac{1}{\omega} \sum_{i=1}^{s^2} diss(J_i(\mathbf{X}_j), \aleph(\mathbf{X}_j)) \,, \qquad (2.16) \\
\text{with} & \\
diss(J_i(\mathbf{X}_j), \aleph(\mathbf{X}_j)) &= \min_k (||J_{NH(i)}(\mathbf{X}_j) - \aleph_{NH(k)}(\mathbf{X}_j)||^2) \\
\text{and } \omega &= s^2(2r+1)^2
\end{aligned}
$$

Here $diss(J_i(\mathbf{X}_j), \aleph(\mathbf{X}_j))$ is simply the sum-of-squared differences between a neighborhood vector NH around pixel position i in $J(\mathbf{X}_j)$ and the best-matching neighborhood in $\aleph(\mathbf{X}_j)$ which was found at pixel k. The size of the neighborhood is defined by its radius r. ω is a normalization factor that makes the measure invariant with respect to image size and neighborhood radius.

As the neighborhood matching employed in calculating the NDM is a very costly procedure, we apply different techniques to speed up the process. We limit the radius of the neighborhood to $r = 2$, resulting in a 5×5-pixel neighborhood which we found to be sufficient (in our test cases). As we use color images, this results in a 75D vector for comparison. We accelerate neighborhood matching by projecting the 5×5-pixel neighborhoods into a truncated 12D principal component analysis (PCA) space. In addition, we use a fast nearest-neighbor search [AMN+94] to find the best-matching neighborhood in $\aleph(\mathbf{X}_j)$.

The NDM has several beneficial properties. The characteristic of the noise function is the total absence of structures, therefore regions in $J(\mathbf{X}_j)$ differing from every neighborhood in $\aleph(\mathbf{X}_j)$ will likely contain some sort of pattern. In addition, this measure also penalizes badly chosen color mappings. Low-contrast images will result in less dissimilarity to $\aleph(\mathbf{X}_j)$, while high-contrast images are preferred.Theoretically, the NDM should be applicable with little changes to other visualization methods, e.g. Pixel Bar Charts [KHDH02]. Such investigations are left for further work, however.

2.7.2 Experiments

As an application example we analyze the *Ozone Level Detection* data set [ZF08] with 2536 instances and 73 dimensions. We test our NDM measure using two different pixel maps: a jigsaw (Figure 2.22) and a spiral-jigsaw map (Figure 2.23). For the jigsaw map, the visualizations for Dim 9 (WSR8), Dim 24 (WSR23), and Dim 60 (U70) are rated to be the worst by our algorithm. They hardly contain any interesting regions and contain mainly noise (Figure 2.22, top row). Dim 9 and Dim 24 depict the measured wind speeds at different times, which are relatively constant with only few changes. On the other hand, Dim 30 (T3), Dim 34 (T7) and Dim 35 (T8), which depict the temperature at 3am, 7am, and 8am in the morning, provide insights into the change of temperature throughout the years.

For the spiral map, Dim 24 (WSR23), Dim 57 (HT85), and Dim 60 (U70) are rated to be the worst according to our algorithm. Similar to the jigsaw map, the worst ranked visualizations contain mainly noise (Figure 2.23, top row), except for Dim 57 that has some structure but is clearly less structured than the best-ranked plots. The two remaining worst views are the same for both methods (Dim 24 and Dim 60). The best rated visualizations are Dim 29 (T2), Dim 30 (T3), Dim 38 (T11) which depict the temperature at 2am, 3am and 11am, respectively. Using different pixel mappings, we can control the local structures presented in the pixel map. The quality measure may be applied to different layouts, and the best results are presented to the user as a guideline for further visual exploration and inspection.

2.8 Discussion

In this chapter we presented several methods to aid in and potentially speed up the visual exploration process for different visualization techniques. We proposed quality metrics to rank four popular visualization methods: scatterplot, parallel coordinates, radviz and pixel-oriented displays, for classified as well as unclassified data. Specifically, we presented new quality metrics for scatterplots for the purpose of finding cor-

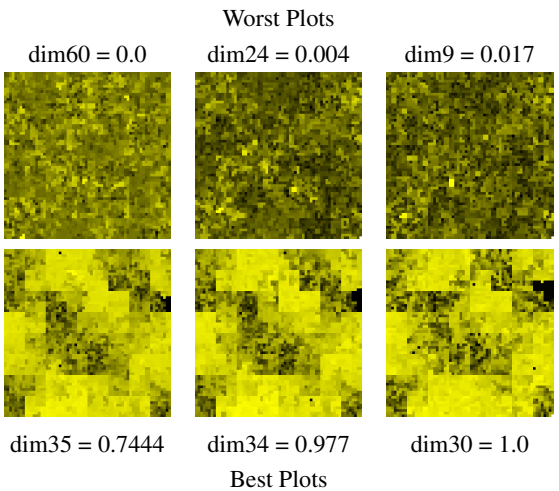


Figure 2.22: Jigsaw maps[Wat05] of the *Ozone* data set [ZF08]: The top row shows the three worst plots, while the bottom row shows the three best plots and their associated normalized goodness values as it was estimated by our NDM. Obviously, the top row contains predominantly noise, and it is difficult to find interesting regions in such a visualization. Also note that its overall appearance is more dull. The best plots according to the NDM show a large degree of potentially meaningful patterns and higher contrast. The goodness values of all plots have been normalized to range between 0 (for the worst plot) and 1 (for the best plot).

relation and cluster separation. For parallel coordinates, we presented quality metrics specialized for the task of finding clusters. Moreover, we presented an improvement for the radviz method with a greedy dimension placement algorithm based on quality metrics. This can be applied to data with pre-defined categorical labels or data sets without any class information. We compare our method with a previously proposed sorting method for radviz [SGM08] and show the improvements introduced by our approach. Finally, we presented quality metrics to detect information-bearing structures in Pixel-Oriented Displays.

Some limitations became apparent in our experiments: Due to a growing number

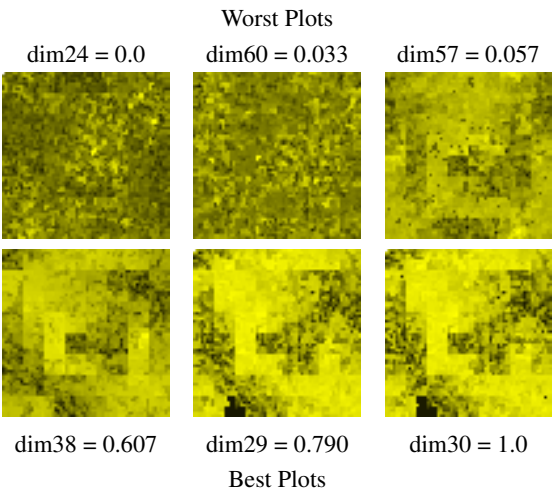


Figure 2.23: Spiral maps of the *Ozone* data set [ZF08]: The top row shows the three worst plots, while the bottom row shows the three best plots and their associated normalized goodness values as it was estimated by our NDM. Obviously,the top row contains predominantly noise, and it is difficult to find interesting regions in such a visualization, except for Dim 57 that has some structure but is clearly less structured than the best-ranked plots. The best plots according to the NDM show a large degree of potentially meaningful patterns and higher contrast. The goodness values of all plots have been normalized to range between 0 (for the worst plot) and 1 (for the best plot).

of classes and due to some multivariate relations it is not always possible to find good separating views. However, this is a general problem and not related to our techniques. Further limitations may, of course, be caused by the task, data complexity, and the metrics applied to find the requested patterns. For example, tasks might be of different types, such as finding outliers, significant patterns, different types of correlations between dimensions etc.

Chapter 3

Perception-Based Visual Quality Metrics

3.1 Motivation

In Chapter 2 we have presented quality metrics that can be used to automatically select promising 2D projections of high-dimensional data. Automatic quality metrics ranking can be applied as a pre-processing step prior to interactive, visual exploration. They can be used to effectively reduce the number of views to be examined by the user, which is done by sorting the views according to their rankings or by selecting the best ones.

Usually defined for some exploration task, the quality metrics can be defined as ranking functions for the projections. However, an open issue that remains is that the range and distribution of ranking values depends on the algorithm of each individual metric. It is not possible to directly compare the results of the different metrics. Specifically, the goodness-of-fit values of the metrics are often relative values, and it is impractical to quantify the amount of structure present in a projection for a specific user task.

In extension to the statistical and mathematical methods discussed in Chapter 2, this chapter presents a quality metric to appraise the quality of a certain projection based on human perception. By means of psychophysics studies, we model the sim-

ilarity between projections in perceptual space and propose a metric based on these observations. Compared to the previously presented methods, this perception-based approach has the advantage that the values assigned to the projections have a direct relation to the perceptual quality as observed by human beings. In other words, we aim at modeling the perception of the visual analyst. The presented metric is very general, can be trained for a variety of exploration tasks, and exploits metrics derived from the human visual system to rank new, unknown visualizations. The contributions presented in this chapter are:

- a new quality metric for projections of high-dimensional data sets based on human perception that is generally applicable;
- can be trained for different visualizations and
- different user tasks;

We present results of evaluating and ranking scatterplots for two exploration tasks: finding correlations between dimensions, and separation between classes. The extension to other visualization methods, e.g. parallel coordinate and pixel-based displays, as well as other user tasks is straightforward. The quality estimation for scatterplots is made based on two psychophysics studies concerning each user task. The first study is used to measure similarity and dissimilarity between visualizations. The second study estimates a ranking based on the user task.

3.2 Related Work

There have been numerous publications to support the exploration of high-dimensional data sets, from the *Projection Pursuit* [FT74, Hub85] method, over the well-known Scagnostics indices [TT85, WAG05], to more recent metrics for different visualization methods and user tasks [SSK06, Guo03, SNLH09, JJ09, TAE+09, AEL+10, DK10]. All these metrics can be used to support the visual analysis of high-dimensional data sets, but they do not necessarily relate to the analyst's opinion.

An initial study towards human perception with visual quality metrics for multidimensional data has been proposed in [TBB+10]. Tatu et al. did a user study to investigate the relationship between human perception and automatically computed metrics. The authors compare three different metrics from [SNLH09, TAE+09] (including the CDM metric presented in Chapter 2) that can be used to estimate class separation in scatterplots and determine which metric better fits the user's opinion. Our approach is different in the sense that we propose an automatic metric based on user perception itself. Specifically, the distance in our ranking function of the scatterplots is optimized to resemble the distance in a perceptual embedding of scatterplots.

In this work we use the analysis of paired comparisons [Dav88] where an individual expresses a preference between two mutually distinct scatterplots. Such comparison studies have been successfully used to produce rankings. Recently, Wills et al. [WAKB09] proposed a method to construct a low-dimensional perceptual embedding for bidirectional reflectance distribution functions that can be used to navigate in the space of gloss and construct new materials. The perception space is built based on a user study with paired comparisons and an extended multidimensional, non-metric scaling algorithm. This multidimensional scaling algorithm copes with incomplete and inconsistent dissimilarity matrices and can be trained using observations from paired comparisons studies. A detailed description of the multidimensional scaling algorithm is presented by Agarwal et al. [AWC+07]. Inspired by this, we construct a similar embedding for scatterplots that, together with a ranking function, can be used as a visual quality metric to quantify the value of the projection, given a specific user task.

3.3 Overview

In this chapter we present a perception-based quality metric that can be used to appraise the quality of visualizations. The goodness value assigned to a visualization by our metric is based on human observations from paired comparison studies. In particular, the numerical differences between metric values are optimized to be consistent with the observed perception distance between the respective visualizations. It can be applied

to a variety of visualizations and exploration tasks. For the presentation in this chapter, we concentrate on scatterplots, and as user task consider two-dimensional correlation search and class separation.

Figure 3.1 gives an overview of our technique. It can be divided into two main phases: a training phase to determine the metric for a specific user task, and a test phase where new visualizations can be ranked using the established metric. The method starts with the training phase. First, a set of visualizations for the specific exploration task is chosen, and a perceptual two-dimensional embedding P for the defined task and visualization is trained based on this set. This embedding provides a perceptually motivated similarity metric that allows to quantitatively compare different visualizations. However, the embedding alone is not enough to decide when a visualization is better suited than another for an exploration task. In order to compare the quality of the visualization, we initialize a second, one-dimensional space R to rank the visualizations. In a final stage, we optimize the distances in R to resemble the distances that were previously established in the perceptional embedding P. This optimized ranking space R is then used to evaluate the quality of new visualizations according to their similarity to the visualizations in the ranking space.

3.4 Perceptual Space for Scatterplots

Our perception-based quality measure is defined based on two user studies with paired comparisons: The first user study serves to estimate mutual perceptual distances between different scatterplot (Section 3.4.1). The second user study is needed to define the ranking function (Section 3.4.2). To train the perceptual embedding and perceptual ranking, a collection of scatterplots S is chosen for each specific user task. The scatterplots can be either synthetically created or taken from the projections of existing high-dimensional data sets. Choosing a representative training set S for the desired user task is of fundamental importance as it directly affects the range of coverage of the final measure.

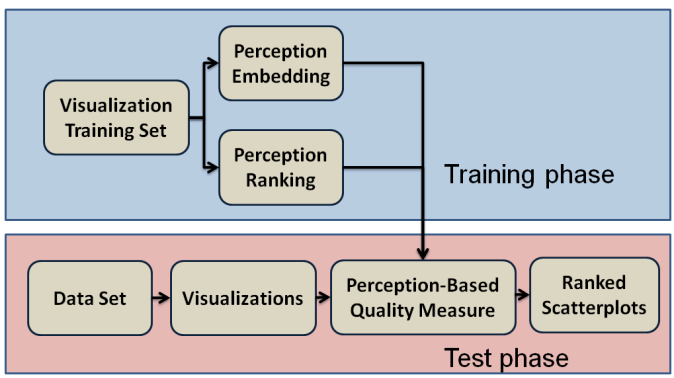


Figure 3.1: Working steps to select the perceptually best visualizations for a given user task. A ranking function is determined in the training phase, based on a perceptual embedding for scatterplots. This function is then used to measure the goodness value of new scatterplots.

3.4.1 Perceptual Embedding

For a specific exploration task, the perception embedding P is built in a two-step approach [WAKB09]: In the first step, a user study is done to estimate similarity between scatterplots. Here, a series of scatterplot triplets is presented to each participant. For each triplet, the participants are asked to decide whether the left or the right image is

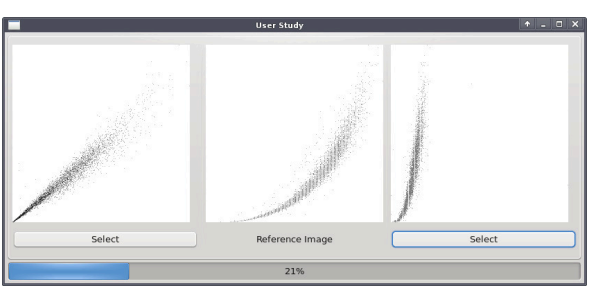


Figure 3.2: Screenshot of the distance comparison test. To each participant of the study, a series of scatterplot triplets is presented. For each triplet, the participant is asked to decide whether the left or the right image is more similar to the central one.

more similar to the central one, according to the specific exploration task (Figure 3.2). The scatterplots are randomly selected from the training set and are different from each other in the triplet. The result of the study is stored as a list of inequalities of the form:

$$L_P = \{(i, j, k) | d_{ij} \leq d_{jk}\}, \tag{3.1}$$

where i, j, k are indices of the scatterplots, and d_{ij} denotes the perceptual distance between the scatterplots i and j. At this stage, we do not know the absolute values of d_{ij} and d_{jk}, but due to the user input we know the correct ordering of i, j, k to fulfill the inequality.

In the second step, the set L_P is used to train the embedding using a general, non-metric multidimensional scaling algorithm (GNMDS) [AWC+07]. Multidimensional scaling (MDS) can be defined as the process of assigning Euclidean coordinates to a set of objects based on a set of constraints. These constraints can be a set of dissimilarities, similarities, or ordinal relations between the objects. The coordinates are assigned to the objects by conserving all constraints as closely as possible. The GNMDS method was developed to learn a low-rank embedding from a collection of paired comparisons, applying convex optimization techniques [AWC+07]. While in the classical MDS the dissimilarity values of the objects are directly interpreted as Euclidean distances, in the GNMDS, only the relative order of the object dissimilarities is necessary. Specifically, the GNMDS method has the advantage that it can be used in a variety of cases where the magnitude of the dissimilarity is uncertain or unknown, as is the case in paired comparison studies. The method can deal with repetitions and inconsistencies that commonly arise in such studies, as distinct participants may have different opinions when comparing the same scatterplots. The evaluation of a set L_P using this algorithm results in a two-dimensional embedding P of the visualizations so that the distances in this embedding directly correlate to the user's perception. In Section 3.6 we show two embedding examples trained according to the user's perception of similarity between scatterplots (Figures 3.8 and 3.9).

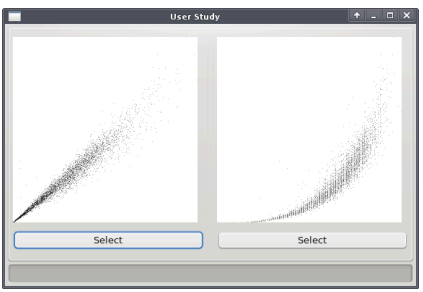


Figure 3.3: Screenshot of the rank comparison test. A series of scatterplot pairs is presented to each participant in the study. For each pair, the participant is asked to decide which of the visualizations has a higher quality considering the specific task.

3.4.2 Perceptual Ranking

Given the perceptual embedding P for the scatterplots, we are able to measure the perceptual distance between them. We now need an efficient way to define a ranking function for the corresponding scatterplots. Our perception ranking R is defined in a three-step approach:

To estimate the ordering of the scatterplots, we perform a second paired comparison study. In this study, two scatterplots are presented to the participants simultaneously. They are asked to choose the best one considering the pre-defined user task. Figure 3.3 shows a screenshot of one such test. Similar to the first study, the presented scatterplot pairs are randomly chosen from the training set, and the result is a collection of observations of the form:

$$L_R = \{(i, j) | p_i < p_j\}, \tag{3.2}$$

where i, j are indices of the scatterplots, p_i represents the quality of the i^{th} scatterplot, and the inequality $p_i < p_j$ denotes that p_j is better then p_i, with respect to the chosen task.

Given the set of inequalities L_R, we need an efficient way to find an optimal ordering for the scatterplots of the training set. Considering that L_R can contain inconsistencies, the problem of finding the optimal ordering for the visualizations is NP-complete

as it is equivalent to the Traveling Salesman Problem [ABCC07]. It therefore requires exhaustively searching all possible visualization arrangements. We use a greedy incremental algorithm to find a suitable order that obeys the observations in L_R as close as possible. Our approach provides a trade-off between finding the optimal solution and completing all computations in a feasible time.

We start by reducing the inconsistencies between the inequalities in L_R. We define v_{ij} as the number of observations of L_R of the form $p_i < p_j$ and $\overline{v_{ij}}$, the number of observations of the form $p_i > p_j$. p_i is considered smaller than p_j only if $v_{ij} - \overline{v_{ij}} > 0$. To insert a new visualization into the current sequence, we test all possible configurations for the new plot within the sequence and choose the one that best fits the observations in L_R. This is done by incrementally placing the actual scatterplot in all possible configurations of the sequence and evaluating how the best ordering fits the observations in L_R. The algorithm is initialized with the first plot, and the best placement for the second plot is computed. We then add the third plot and create all possible sequence arrangements. The process is repeated, inserting each new element one by one at the position that fulfills the most inequalities in R, until all elements have been added. The value of a sequence is defined by the number of observations that are fulfilled and is defined by:

$$s = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \begin{cases} 1, & p_i < p_j \\ 0, & otherwise \end{cases} \tag{3.3}$$

When all scatterplots are added, the sequence with highest value s, i.e. the largest number of fulfilled observations, is chosen as the relative ordering for the scatterplots. Figure 3.4 shows an illustrative example for three visualizations. Given the set of observations $p_1 > p_2$, $p_1 > p_3$ and $p_2 < p_3$, the best placement for p_2 is found (Figure 3.4c) and the final sequence is defined (Figure 3.4e).

Until now the ordering is only relative. In order to quantitatively evaluate the scatterplots, we need to combine it with the perceptual embedding P. Our goal is to find a ranking that, while constrained by the optimized ordering, fits the mutual distances in P as close as possible. Figure 3.5 depicts such an example. Given an ordering of

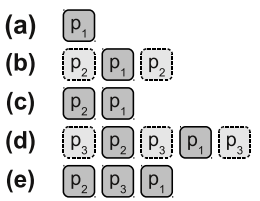
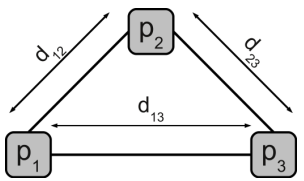


Figure 3.4: Placement algorithm for three example visualizations given the set of observations $p_1 > p_2$, $p_1 > p_3$ and $p_2 < p_3$. (a) The algorithm is initialized with the first visualization p_1. (b) All possible sequence arrangements for p_2 are tested. The set of possible insertion positions are marked with dotted boxes. (c) The best sequence is determined and saved. (d) The algorithm searches for the best placement for p_3, and the best sequence is determined (e).

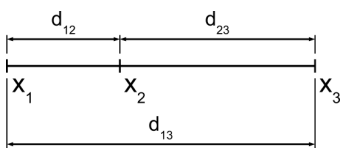
the visualization in the form $p_1 \leq p_2 \leq p_3$, determined by the algorithm in Figure 3.4, and their mutual distances in the perceptual embedding d_{12}, d_{13}, and d_{23}, we want to find the best fitting goodness values for x_1, x_2, x_3, where x_i is the rank value for the scatterplot p_i. Stating the problem in terms of a matrix $A\mathbf{x} = \mathbf{d}$, we want to solve the following optimization problem:

$$\begin{aligned} E = \arg\min_{\mathbf{x}} \|A\mathbf{x} \quad &- \quad \mathbf{d}\|_2^2 \\ x_i \quad &\leq \quad x_j,\ i < j. \end{aligned} \tag{3.4}$$

To solve this problem, $\mathbf{x}$ is first initialized with the cumulative pairwise distances of the embedding P, i.e. the scatterplot ranked lowest is initialized with $x_1 = 0$ and each remaining scatterplot with $x_i = x_{i-1} + d_{i,i-1}$. Afterwards, the values x_i are optimized to fit as close as possible all distances d_{ij} of the embedding P. The minimization of Equation (3.4) can then be computed using quadratic programming. As the problem is convex we iterate several times over $\mathbf{x}$, optimizing each point locally, until the algorithm converges to the global minimum.



(a) Perceptual Embedding



(b) Perceptual Ranking

Figure 3.5: Embedding and ranking examples: given an ordering of the visualizations $p_1 \leq p_2 \leq p_3$, determined by the algorithm in Figure 3.4, and their mutual distances in the perceptual embedding d_{12}, d_{13}, and d_{23} we want to find the best values for x_1, x_2, x_3.

3.5 Perceptual Query

After defining the perceptual ranking for a chosen user task, we can finally appraise the quality of new scatterplots. To define the goodness value of a new scatterplot p_q, we search for the k-nearest scatterplots in our training set. Instead of directly comparing points of scatterplots, we choose to extract specific features to represent them and yield a more robust comparison. Principal Component Analysis (PCA) [TP91] is a common approach to find such robust feature descriptors of images and can be used for scatterplot images as well. We follow this approach by using the set of previously selected scatterplots in the training phase to compute an eigenobject basis. After projecting the scatterplots onto this eigenobject basis, the first ten main components are chosen as feature vector.

For a new query we use a fast nearest-neighbor search [AMN+94] to find the k-best matching scatterplots in our training basis. If k is larger than 1, the influence of outliers is reduced. We use $k = 3$ throughout the chapter. The final goodness value is computed as the weighted sum of the goodness values of the k best matching scatterplots. The

Perception-based Measure (PBM) is therefore defined as:

$$PBM = \frac{\sum_{k=1}^{K} w_k x_k}{\sum_{k=1}^{K} w_k}, \tag{3.5}$$

where the weight w_k is defined as $\frac{1}{d_k}$, with d_k being the Euclidean distance between the feature vectors of the query and the k^{th} best scatterplot. It is worth noting that choosing a representative training set for the desired user task is of fundamental importance for this method because the final measure value for a new scatterplot is computed based on the values of the most similar plots in the training set.

3.6 Experiments

Our PBM (Section 3.5) can be trained to select high quality projections for different user tasks. For each user task, the metric is trained using a set of training visualizations to represent the task. The quality of the training set is crucial to the outcome of the PBM as it is always relative to the provided set. In the best case, this training set should contain evenly distributed examples of the possible space of all visualizations as the later goodness value of the visualizations is dependent on this training set. But even with incomplete training data, the results are often found to be sufficient. For the tests presented in this chapter we only used incomplete training data.

We trained our metric separately for two distinct exploration tasks: correlation-finding for unclassified data, and class separation for classified data, which are two standard tasks in visual analytics [SNLH09, JJ09, TAE+09]. For the correlation analysis task, we have chosen a set of scatterplots from the abalones data set [Nas94], containing examples of linear and non-linear correlation, as well as different degrees of correlation. Figure 3.6 shows three examples of 21 scatterplots chosen as training set to represent this task. For the class separation task, we have chosen a set of scatterplots from a synthetically generated data set [TAE+11], containing examples with two distinct classes represented by two point clouds in two different colors, Figure 3.7. By choosing an example with only two classes, we aim to show the effectiveness of

our metric for the class separation task. Note that the class separation task consists of finding scatterplots were the classes, which are already known and represented by different colors, are well separated. However, our metric can be similarly trained for the task of finding projections containing separate clusters in data sets where no label information is available. Figure 3.7 shows three examples of 28 scatterplots chosen to represent this task. These training sets are not complete to describe the respective tasks. They nevertheless proved sufficient to successfully show the effectiveness of the approach.

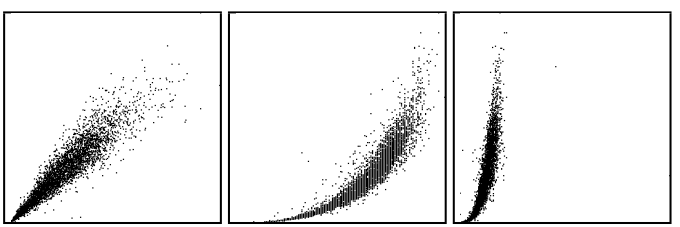


Figure 3.6: 3 Examples of 21 scatterplots from the training set for the correlation task.

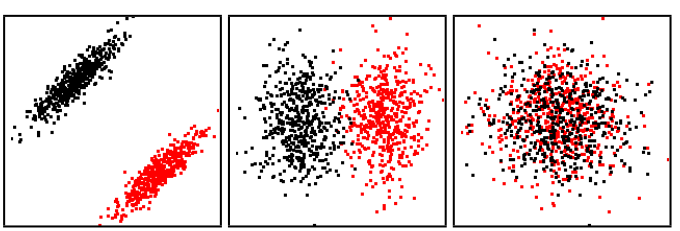


Figure 3.7: 3 Examples of 28 scatterplots from the training set for the class separation task.

For each task, we conducted two psychophysics studies using the selected scatterplots. The participants of the studies were 20 undergraduate students, PhD students and postdocs working in the field of computer graphics. In the first study, 200 randomly chosen scatterplot triplets from the task training set were presented. The users were then asked whether the left or the right plot is more similar to the central one, Figure

3.2. In the second study, 100 scatterplot pairs were presented to the participants, again randomly chosen from the task training set, and the participants were asked to choose the best scatterplot considering the given task: For the correlation task, the participants were instructed to observe the amount of correlation between the scatterplot axes and the difference between linear and non-linear correlation. Similarly, for the class separation task, the users were asked to pay attention to the separation between the class clusters, i.e. the best plots show well-separated clusters and no overlap between the classes.

3.6.1 Perceptual Space

Using the results of the first user study, we train two perceptual embeddings as described in Section 3.4.1. Figure 3.8 shows the resulting two-dimensional perceptual embedding trained using the scatterplots from the training set for the correlation task. It is worth noting how the scatterplots are clustered according to the *kind* of pattern (linear and non-linear) and the amount of correlation. We can clearly see three main clusters: scatterplots presenting high and non-linear correlation clusters on the right of the embedding, lower non-linear correlation at the top, and scatterplots depicting nearly linear correlation can be found at the bottom left. Additionally, we can observe three important sub-classifications in this last cluster: strong linear correlation in the middle, sparse linear correlation on top, and lower linear correlation on the bottom.

Figure 3.9 shows the resulting perceptual embedding for the class separation task. Similar to the correlation space, the clusters of scatterplots can be observed according to the presented class separation. Three main clusters can be observed in this embedding as well: scatterplots with bad separation between the two classes on the right, with horizontal separation in the top-left corner and with vertical separation in the bottom-left corner.

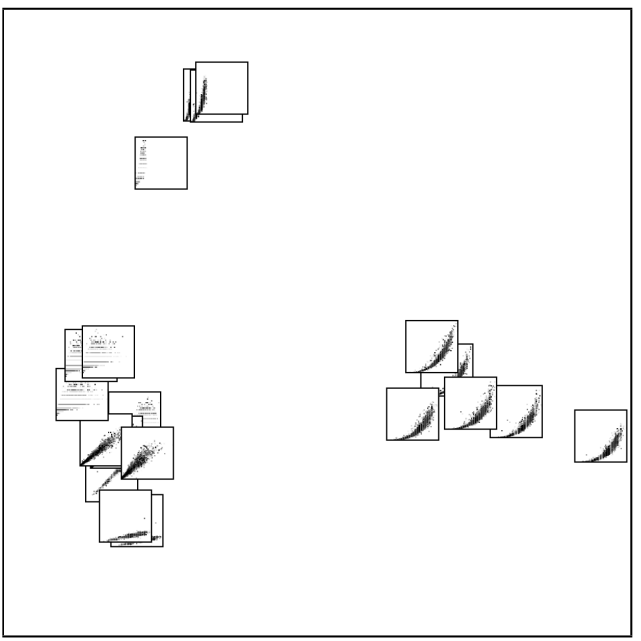


Figure 3.8: Resulting two-dimensional perceptual embedding trained using scatterplots for the correlation task. The scatterplots are clustered according to the *kind* of axis correlation (linear and non-linear) and the amount of correlation. We can see three main clusters in this embedding: scatterplots presenting high and non-linear correlation cluster at the right of the embedding; lower non-linear correlation at the top, and scatterplots depicting nearly linear correlation can be found at the bottom left.

3.6.2 Perceptual Ranking

The perceptual rankings for the respective tasks are created based on the second study as described in Section 3.4.2. Figure 3.10 shows the resulting ranking for the correlation task. Small-scaled scatterplots are shown in the middle row, and their respective quality values are show in the bottom row. Note that as a result of the optimization, similar plots have the same or almost the same quality value. This value increases according to the distance between the plots in the perceptual embedding (Figure 3.8). A

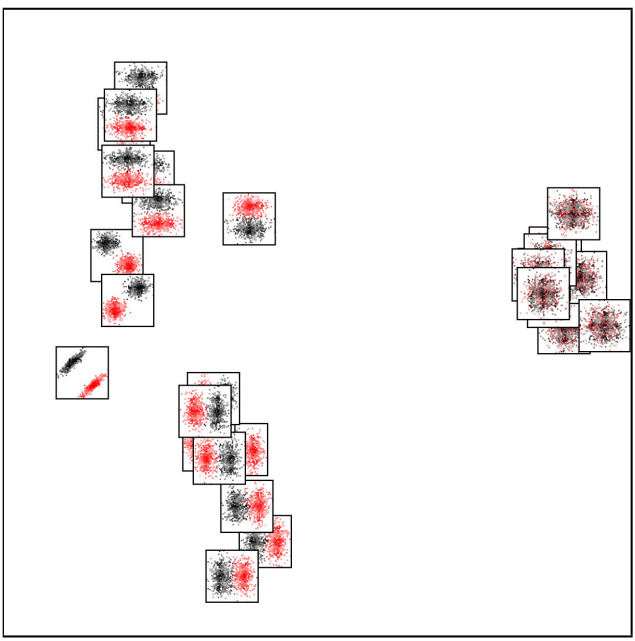


Figure 3.9: Perceptual embedding for the class separation task. Three main clusters can be observed: scatterplots with bad separation between the two classes on the right; a cluster of scatterplots with horizontal separation in the top-left corner, and scatterplots with vertical separation in the bottom-left corner.

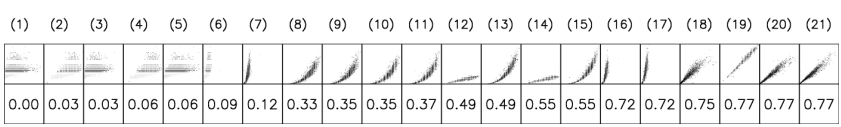


Figure 3.10: Perceptual ranking for the correlation task, top row. The scatterplots are shown in the middle row, and their respective quality values are show in the bottom row. Note that similar scatterplots have the same, or almost the same quality value. This value increases with their mutual distances in the perceptual embedding (Figure 3.8).

larger discrepancy between values can be observed when neighboring scatterplots are quite different due to the different clusters in the perception embedding. Examples in

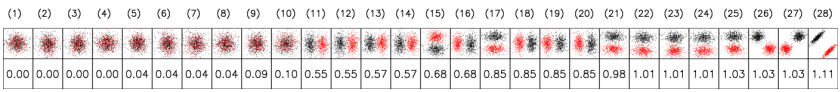


Figure 3.11: Perceptual ranking for the class separation task. We observe large discrepancies between the quality values of differing scatterplots, e.g., between scatterplot (10) and (11) where the separation of the two classes differs distinctly.

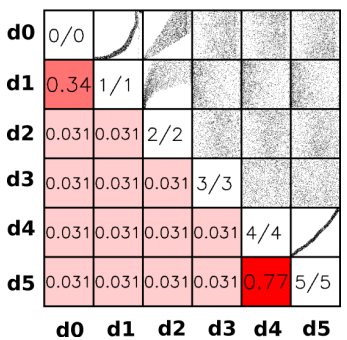
Figure 3.10 are between scatterplots (7) and (8) where (7) behaves as an outlier, and between (15) and (16) where the transition from non-linear to linear correlation occurs. The resulting order of the scatterplots resembles the observations of the second study. One characteristic that can be observed in the correlation ranking is the preference of the participants for scatterplots that present linear correlation between dimensions.

Some participants were unexperienced in the field of visualization and visual analytics. These participants had problems choosing scatterplots showing higher correlation value as they assumed correlation and skinny structures to be equal. This resulted in outliers that can be minimized by using more experienced participants for the study.

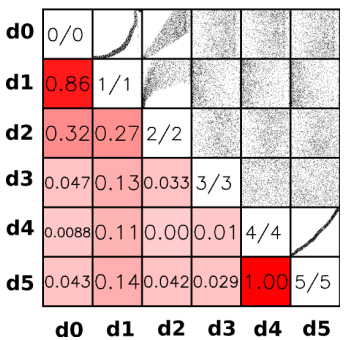
Figure 3.11 shows the resulting ranking for the class separation task. Again, we observe large discrepancies between the quality values of different scatterplots. One example is between scatterplot (10) and (11) where a clear difference concerning the separation of the two classes can be observed. From (1) to (10), the scatterplots present a large overlap between the classes. An interesting aspect that can be noted is the preference for horizontally separated class clusters. During the study we could observe that horizontally separated point clusters appeared perceptually to be more disjointed to the participants than vertically separated clusters.

To verify the stability of the scatterplot ranking, we performed a leave-one-out test with the scatterplots of the class separation task. We removed each of the 28 scatterplots and trained the metric with the 27 remaining samples. The newly created rankings presented an average difference of 1.28 position compared to the original one. This corresponds to an error of 4.6%, indicating that the ranking is stable considering the dependence on the individual scatterplots of the training set.

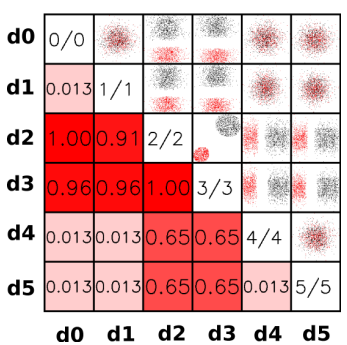
3.6.3 Perception-Based Metric



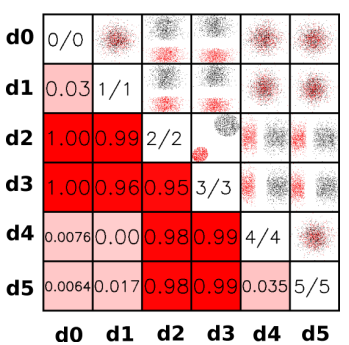
(a) Perception-Based Metric - Correlation Task



(b) RVM Metric - Correlation Task



(c) Perception-Based Metric - Class Separation Task



(d) CDM Metric - Class Separation Task

Figure 3.12: Scatterplot matrix of two test data sets, showing scatterplots above the main diagonal and their respective values according to the metrics under the diagonal. (a) Results for the PBM metric and the correlation task, scatterplots with high correlation between the dimensions present a high quality value. (b) Results for the RVM metric. (c) Results for the PBM metric and the class separation task, scatterplots with well separated classes present a high quality value. (d) Results for the CDM metric.

To evaluate our metrics we test them with two synthetically generated data sets. To test our PBM for the correlation task, we use an unclassified data set with 6 dimensions

and 1000 sampling points. Figure 3.12a presents the scatterplot matrix of the data set, showing scatterplots above the main diagonal and the respective PBM values below the diagonal. These values are computed based on the 3-best matching scatterplots of the training set. E.g. for the scatterplot (Dim 0 - Dim 1), the scatterplots (10),(11), and (15) are automatically chosen from the training set (Figure 3.10) and their respective ranks are used to compute the PBM (Section 3.5). Our metric successfully ranks linear and non-linear correlation between dimensions (Dim 0 - Dim 1) and (Dim 4, Dim 5) with suitable values according to the previously computed perceptual ranking. The remaining scatterplots are ranked with a low value for the PBM , consistent with the perceptual ranking.

Similarly, we compute the values of the scatterplots using the RVM described in Chapter 2 (Figure 3.12b). The RVM is used to find linear and non-linear correlations between pairwise dimensions. Both metrics successfully select the scatterplots with the strongest correlation as best plots. Note that in the perception ranking we train for correlation (Figure 3.10), the worst-ranked scatterplots are similar to the scatterplots (Dim 0 - Dim 2) and (Dim 1 - Dim 2), justifying their low values for the PBM. One solution to achieve better sensitivity between the worst-ranked plots is to expand the training set for this task. Compared to the RVM , the PBM has the advantage that the ranking values resemble the user perception. Furthermore, it can be trained not only for correlation but for a variety of other user tasks. The PBM has a lower sensitivity due to its restriction to the used training set. Using a more comprehensive training set may increase this sensitivity.

To test the PBM with the class separation task, we use a classified data set with 6 dimensions, 2000 sampling points and two classes. Figure 3.12c depicts the scatterplot matrix of this data set showing scatterplots above the main diagonal and the respective PBM values bellow the diagonal. The PBM successfully ranks the new scatterplots according to the previously defined perceptual ranking (Figure 3.11). Note that the scatterplots with horizontal class separation are rated higher than scatterplots with vertical class separation. Apparently, user prefer horizontal separation over vertical separation of clusters.

Figure 3.12d shows the CDM from Chapter 2 computed for this data set. The CDM evaluates scatterplots according to the separation properties of the classes. Comparing our PBM to the CDM , we observe that the PBM yields more accurate values. For example, the scatterplot (Dim 2 - Dim 3), that presents the best separation of the class clusters, is ranked with the highest value by the PBM but not by the CDM.

3.7 Discussion

In this chapter we have presented the first perceptually-motivated quality metric for evaluating visualizations. Our new quality metric serves several purposes. Since it is a very general technique, based only on the user opinion caught from psychophysics studies, it is not limited to scatterplots. We expect it to also be applicable to other visualization methods such as parallel coordinates or Pixel Bar Charts. We showed how to apply our technique to typical visual analytics tasks such as finding projections with high (non-)linear correlations or good class separability. Other user tasks can be addressed in the same fashion. We compared our perception-based method with the quality metrics introduced in Chapter 2 and showed that this new approach can be used to rank visualizations. Our perception-based quality metric has the advantage that it can be trained to evaluate a variety of user tasks. The computed quality values resemble human perception, even though there are limitations due to the training set and user preferences.

The presented metric opens up new possibilities to aid in the visual exploration of high-dimensional data sets by capitalizing on the human visual system.

Chapter 4

Visual Quality Metrics Visualizations

4.1 Motivation

With the exponentially increasing amount of multivariate data, several multidimensional visualization techniques have been proposed during the last decades [Kei02]. Based on the fact that human perception cannot deal well with more than two or three continuous dimensions simultaneously, most techniques project the data in low-dimensional embeddings and combine these representations in a single plot or present them to the user in an interactive way. Some well-known examples of multidimensional visualization techniques are parallel coordinates [Ins85], scatterplot matrices [Har75], glyph techniques [War94] and pixel level visualizations [KAK95]. But even these techniques do not scale well to high-dimensional data sets. In this chapter we focus on parallel coordinates (PC) and scatterplot matrices (SPLOM), and propose some extensions to these well-known visualization techniques to cope with high-dimensional data: a class-based scatterplot matrix and an importance-oriented reordering of the dimensions of the matrix. The rationale behind the *class-based SPLOM* (C-SPLOM) is to support the visual analysis of labeled (classified) data sets. In such data sets, the analyst often searches for projections of the data that depict distinct clusters. Previous approaches, such as the quality metrics presented in Chapter 2, aim at finding good views of a data set considering all classes at once. The problem with such approaches

is that global optimization may ignore views that separate only two classes well, because of the distribution of the remaining classes. Our C-SPLOM approach presents the best projection for each class pair, based on a ranking index. This class-based visualization method is useful to analyze labeled data sets with a large number of variables which cannot be well visualized using traditional SPLOMs.

Similar to the scatterplot matrices, parallel coordinates do not scale well when the number of dimensions grows as important multidimensional relationships might not be visualized. Addressing this shortcoming, we propose an importance-oriented *parallel coordinates matrix* (PCM). Unlike the SPLOM the PCM is not symmetric. Each row i of the matrix represents the relation of one dimension d to all others, ordered by a quality metric ranking. Additionally, we propose a quality aware dimension reordering framework for visualization matrices, like SPLOMs, C-SPLOMs and PCMs, to improve the visual analysis task of high-dimensional data sets.

Another important issue in visual exploration concerns the case where too many different data values must be represented in one visualization. A common approach in many of the existing techniques is to represent the value of a variable using color scales, e.g. in choropleth maps [Dup26, Wri38], or pixel-oriented visualizations techniques [Kei00, Wat05]. The mapping of data values onto this color map is of crucial importance for the visual analysis task as it can hide or reveal important structures. In classical data exploration, playing with different mapping functions, also called transformations, to find a promising color mapping is an important part of the exploration process but it can be time consuming as there are essentially infinite possibilities.

Many visualization and statistical methods assume that variables are normally distributed. However, real data of a variety of fields present a non-normal behavior[Mic89]. There is a great variety of transformations that are used to improve normality of variables, e.g. adding or multiplying constants, taking the square root or converting to logarithmic scales. The logarithmic transformation is discussed in [Cle84] as a very powerful tool for graphical representations. It can be used to transform variables that are right-skewed and generates pleasing visualizations of an otherwise bad resolution distribution. In such visualizations, a few large values take up most of the color map

scale, and the rest of the data points are squashed into a small part of the scale with low resolution. Nonetheless, it is worth noting that in practice the analyst does not know a-priori which normalization function is best-suited for the data.

In this chapter we present a simple, yet effective and automatic data transformation method to guide the color mapping of a visualization based on the underlying data. The data values are transformed into a joint two-dimensional space, where the y-axis depicts the data value while distances on the x-axis represent the influence of each data point. By re-projecting these vectors onto a one-dimensional space, spanned by the smallest and largest data value, we successfully improve discriminability of the resulting visualization and potentially increase its information content, without resorting to a-priory knowledge about data distribution. We provide the user with a simple-to-use interpolation technique to continuously change the color mapping from a transfer function where each data value is granted the same amount of space on the color map, to a linear transfer function which enhances outliers.

4.2 Related Work

SPLOM and PCP are two of the most popular multidimensional visualization techniques and are implemented in diverse popular visualization tools, for example, in the XmdvTool [War94] and GGobi [STBC03].

The SPLOM was first published by John Hartigan [Har75] and later explored and extended in diverse visual exploration tools. Unfortunately, SPLOMs lose their effectiveness if the number of dimensions becomes too large. To deal with this problem, different approaches have been proposed, e.g projection pursuit [FT74, Hub85, WAG05] or our metrics discussed in Chapters 2 and 3. For the methods presented in this chapter, we make use of such metrics in a twofold way, to select information-bearing projections for the C-SPLOM and PCM, and to perform dimension reordering. In case of classified data sets, a class consistency visualization algorithm has been proposed by [SNLH09]. Similar to our class based matrix, the class consistency method proposes measures to rank lower dimension representations. The method proposed in [SNLH09]

filters the best scatterplots based on their ranking values and presents them in an ordinary scatterplot matrix. One problem of this method is that the SPLOM does not scale well for high-dimensional data sets, and even if a zoom option is available, the overall visualization of the SPLOM is prejudiced. Another problem occurs if all classes are analyzed together to rank the projections. In this case, projections that separate two classes very well might receive a bad ranking because of the distribution of the remaining classes. Our method reduces the matrix size to the number of classes of the data set and presents the best projections for each class pair to the user individually.

Parallel coordinates [Ins85] is another very popular multivariate visualization technique. In parallel coordinates, each dimension appears just once, and the relation with other dimensions may be difficult to pinpoint depending on the distance between them in the plot. Diverse linking and brushing algorithms [War94, STBC03], together with transparency levels, have been proposed to help visualizing these relations. However, they do not solve the problem if one dimension shares important correlations with more than its two neighboring dimensions in the visualization. We propose a parallel coordinate matrix where there is the possibility to plot all possible three-dimensional combinations for each dimension. In this matrix for each dimension d of the data set, we have up to $(n-1)/2$ three-dimensional parallel plots, where d is the central dimension. Theoretically, these plots reveal most important relations between this central dimension and the others. An important issue for parallel coordinates is how to order the dimensions in the plot. Different proposals to solve this problem focus on ordering the dimensions by similarity [ABK98, YWRH03]. Our method discussed in Chapter 2 proposes a sorting of the dimensions based on the quality of the plots. A ranking function evaluates each two-dimensional parallel plot. The result is used to determine the order of the dimensions in the final plot. Our PCM capitalizes on this approach to order the three-dimensional individual plots. For each dimension, we sort its respective three-dimensional plots using a ranking function; the plots with a higher ranking are presented first, and the ones with a lesser amount of useful information are presented last.

Some specialized methods for color mapping can be found in the literature, e.g. May et al. [MDK10] who use a truncated linear scaling combined with a sign test to compute a signed difference to distinguish profiles whose frequencies lie above or below an expected distribution, but these are not generally applicable to arbitrary data sets. The Pixnostics method, presented in [SSK06], investigates the importance of choosing an appropriate set of parameters for pixel-oriented visualizations, including a convenient color mapping, but does not propose how to choose a useful parameter set for the transformation beforehand. Borland et al. [BI07] propose to choose the color map itself based on the underlying data. They state that no automatic method yet exists, which can establish the optimal color map automatically.

4.3 Overview

In the following sections we define our information-bearing visualization matrices in more detail and describe the quality metrics we use to rank the low-dimensional projections. Section 4.4 describes technical details about our *parallel coordinates matrix* and show some experiments using real, high-dimensional data sets. Section 4.5 presents our *class-based scatterplot matrix* and shows how it can be used to support the visualization of classified data sets. In Section 4.6 we discuss reordering of scatterplot matrices using such quality metrics and how it can help to visualize high-dimensional data sets. Finally, in Section 4.7, we present our content-aware color mapping algorithm that can be used to transform data set values of different distributions to reveal interesting structures in the data.

4.4 Parallel Coordinates Matrix

Parallel coordinates [Ins85] are one of the techniques which allow visualizing an arbitrary number of dimensions of a data set within the same plot. This makes them very attractive for high-dimensional data sets but comes at a cost. The amount of information-bearing content is very sensitive to the ordering of dimensions [ABK98].

In addition, every dimension can be paired with only two other dimensions. Therefore, important relations to a third, fourth or more dimensions might be missed.

Our approach aims at presenting all those relations in a single matrix where each entry shows the relationship between only two dimensions. This way, all n^2 possible combinations of dimensions are represented and no information is lost. The problem with such an approach is potential a overstraining of the user as he would have to check every single visualization for possible information content. It is therefore important to sort the visualizations inter- as well as intradimensionally, so that important visualizations are spatially close together in the matrix. Such a matrix is most legible if three constraints are fulfilled:

1. Every row should contain one primary dimension which appears in every visualization in this row. A label is assigned at the left of the row for faster indexing.

2. The visualizations in each row should be sorted in descending order according to their inherent information value. The best should be positioned on the left, the worst on the right.

3. The dimensions itself, i.e. the rows of the matrix, should be rearranged so that the rows with the most valuable visualizations are on top, while dimensions with less information value are closer to the bottom of the matrix.

This way, only looking at the $n \times p$ submatrix, starting at index $(0,0)$ reveals the potentially most valuable relationships, i.e. visualizations, to the user. An example of this concept is given in Figure 4.1.

In a first step, all n^2 2D visualizations are created. A quality metric is applied to assess the possible informational value of each visualization. Most approaches in the literature aim at finding the best ordering of all n dimensions globally, or selecting a subset of them. In Chapter 2 we presented an approach that rates every parallel coordinates plot consisting of only two dimensions and combines them in a second step to the complete visualization. For our test data, we exemplarily make use of the *Overlap Measure* (Section 2.5.2), which measures the overlap between different

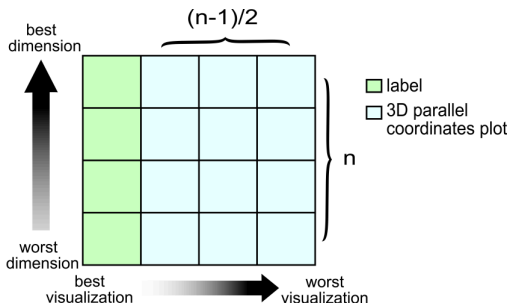


Figure 4.1: Structural overview of the PCM. Each row has one primary dimension, appearing in the middle of each 3D plot and as a label on the left for better overview. The rows are ordered according to the overall importance of the primary dimension, from left to right in descending order. The visualizations in each row are again ordered according to their relative importance.

classes of the data set in Hough space. Visualizations with distinct classes receive a high quality value, and visualizations with very similar classes receive a low value. Other quality metrics, for class and non-class based data sets, can be easily included in our framework.

We initialize the matrix so that each row of the matrix has one primary dimension, e.g. each visualization in row 1 contains dimension d_1, each visualization in row 2 dimension d_2 and so on. We then sort the visualizations intra-dimensionally, i.e. per row. As each visualization is associated with a quality value, we can easily apply a simple standard sorting algorithm. We always combine two two-dimensional visualizations to a three-dimensional visualization as both share the same primary dimension, which is then positioned in the middle.

In a final step, we re-order the dimensions, i.e. the rows of the matrix. We tested different criteria, like summation of all quality values in each row, or linear and Gaussian falloffs, increasing the importance of the first visualizations in each row while decreasing the importance of the lesser-valued ones. We found that the linear falloff gives good results for the PCM. More details and a more general description for dimension-reordering visualization matrices is given in Section 4.6. The quality value of the j-th

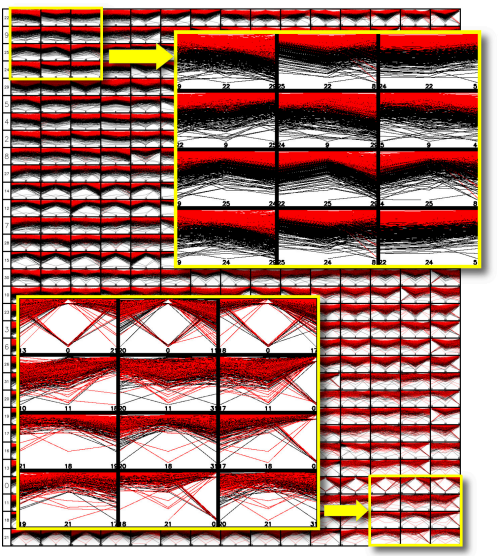


Figure 4.2: Results of the PCM for the WDBC data set. Malign cell nuclei are colored black while healthy cell nuclei are red. Visualizations with only little overlap are preferred, so the difference between malign and benign cells becomes clearer. These can be found in the top left of the matrix. The worse visualizations on the bottom right hardly convey any useful information.

dimension is computed by

$$D_j = \sum_{i}^{n} \frac{(n-i)}{n} Q(p_{(j,i)}) \quad , \tag{4.1}$$

where n is the number of dimensions and $Q(p_{(j,i)})$ is the quality value for the i-th visualization in the j-th row of the matrix.

4.4.1 Experiments

We use the *Wisconsin Diagnostic Breast Cancer* data set (WDBC) to test the usefulness of our PCM. The WDBC data set consists of 569 samples with 30 real-valued

dimensions each [SWM93]. The task is to find the best dimensions separating the malign and benign cells in the data set. We created our PCM for this data set using the *Overlap Measure* from Chapter 2. Other metrics could be used as well, depending on the task. Figure 4.2 shows the complete PCM with the best and worst-ranked visualizations enlarged. Visualizations with higher information content are found in the top left of the matrix, as intended, while the visualizations on the bottom right are hardly of any use. Looking only at the best parallel coordinates plots, as in [YWRH03], one might miss important information. For example, dimension 22 (radius (worst)) in combination with dimension 9 (concave points (mean)), 29 (concave points(worst)), 25 (area (worst)) and 5 (area (mean)) all separate the malign and benign cells comparatively well, but in usual parallel coordinate visualizations only two such combinations can be displayed in one visualization.

One could argue that SPLOMs fulfill a similar task as PCMs, but there are major differences between these two approaches. First, SPLOMs are not sorted. This limits their usefulness to data sets of up to only a dozen dimensions. Beyond that, the exhaustive investigation of each plot is taxing the user. Even when sorting the dimensions beforehand, as proposed in Section 4.6, additional information, such as color encoding or ranking values, are needed to guide the visual search. Using PCMs, looking at the $n \times p$ sub-matrix starting at index $(0,0)$ may reveal the most valuable relationships to the user, no matter what the dimensionality of the data set is. Of course, the choice of parallel coordinates can also be exchanged for scatterplots. Which one is more beneficial depends on the familiarity of the user with the approaches.

4.5 Class-Based Scatterplot Matrix

A common task in visual analytics is to search for projections of high-dimensional data that show well-defined clusters. The same occurs when class information is available; finding the projections or dimensions that can separate the distinct classes well is a desired outcome. To serve this purpose, we introduce a new visualization matrix called *class-based scatterplot matrix* (C-SPLOM). We assume that each point in the

high-dimensional space has a class label c. Diverse data sets have a clear definition of classes. But, since class labels can be assigned through an automatic clustering algorithm this technique is not limited to class-based data sets.

Similar to the well-known SPLOM, the class-based version is also a matrix of pairwise scatterplots $s(a,b)$ with data dimensions a and b. The difference is that instead of the original dimensions, the classes are listed as rows and columns. If there are m classes in a data set, the C-SPLOM has dimensions $m \times m$ and the element at the i-th row and j-th column is the scatterplot of the k-th and h-th variable. The projection axes k and h are chosen in such a way as to maximize the information content for the pairwise relation of the i-th and j-th classes.

An important issue of the C-SPLOM is to choose an appropriate analysis algorithm to compute the quality index $Q(s(a,b))$ of the scatterplots. Different algorithms can be used to this end, e.g. the quality metrics presented in Chapter 2, as long as they consider the pairwise relationships between classes. The problem in considering all classes at once, as proposed in Section 2.4.2, is that the global optimization may ignore views that separate two classes well because of the distribution of the remaining classes. Figure 4.3 shows examples of scatterplots generated from the *Olives* data set (Section 4.5.1). Considering all classes, the first scatterplot $s(4,5)$ has the highest rank $Q(s(4,5)) = 1$. However, the scatterplot $s(2,8)$ with rank $Q(s(2,8)) = 0.44$ presents a better separation of the 3-th and 4-th class present in the data set (*South-Apulia* region in red and *Sicily* region in green, respectively), as can be seen in the third plot. This observation is only possible if the adopted metric analyzes the pairwise relationships between classes instead of a global metric. The resulting quality index Q is then used to rank the scatterplots, and the best scatterplot is selected for the respective class pair.

We test our C-SPLOM with two similar algorithms to measure the quality of scatterplots with class information. The first algorithm is the *Class Density Measure* (CDM) presented in Section 2.4.2. It assigns high values to plots with few overlap between classes and dense clusters. The second metric is the *Class Separating Measure* (CSM) (Section 2.4.2). As the CDM, it yields high quality values for plots with

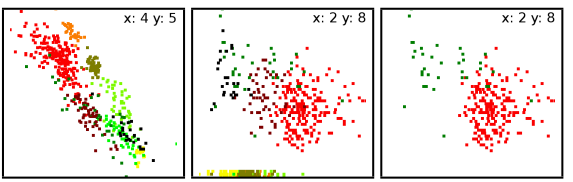


Figure 4.3: The first scatterplot (left) is the one with the highest rank $Q = 1$. Considering all classes, however, the second one with rank $Q = 0.44$ (middle) presents a better separation of the 3-th and 4-th class (red and green), as can be seen in the third plot (right).

well-separated clusters. Instead of dense clusters, however, this metric has a bias towards larger distances between clusters. We adopt the CDM and the CSM algorithms with the difference that only one class pair is considered at a time. To rank projections considering a specific class pair, the algorithms are applied only to the data of the respective classes, and the best-ranked scatterplot represents this class pair in the C-SPLOM. To decide which algorithm is the best one depends strongly on user task. Figure 4.4 shows an example of the differences between the C-SPLOMs for the *Wine* data set (Section 4.5.1) using these two approaches.

Note that for the 1-st and 2-nd class (in black and red, respectively) the CDM yields a scatterplot with more dense clusters as best result, while the CSM yields a scatterplot where the distance between the center of the clusters is larger. The same happens for the 1-st and 3-rd class (in black and green, respectively). For the 2-nd and 3-rd class the same scatterplot is chosen by both metrics.

4.5.1 Experiments

To evaluate our C-SPLOM, we test it on different real data sets from the UCI repository [BL13] with labeled information. The first presented data set is the *Wine* data set, a classified data set with 178 instances and 13 attributes describing chemical properties of Italian wines derived from three different cultivars (classes). The user task here is to find the projections (dimensions) that separate these three classes well. Figure 4.5 shows the comparison of the C-SPLOM (upper-right) and its counterpart SPLOM

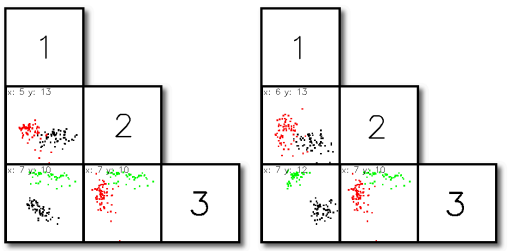


Figure 4.4: The resulting C-SPLOM for the CDM (left) and CSM quality measures (right).

(bottom-left). The C-SPLOM is computed by means of the *class separating measure* described previously. Another data set we use to evaluate the C-SPLOM is the *Olives* [ZNLG94] data set, with 572 olive oil samples from nine different regions of Italy. For each sample the normalized concentrations of eight fatty acids are given. Figure 4.3 show two scatterplots of this data set, the first one with the 4-th and 5-th dimensions (concentrations of the oleic and linoleic acids), and the second with the 2-nd and 8-th dimensions (concentrations of the palmitoleic and eicosenoic acids).

4.6 Dimension Reordering

Often, *n*-dimensional data sets are represented as a series of two-dimensional scatterplots. Such scatterplots are commonly arranged in a SPLOM, and usually the dimensions are arranged in the same order as provided by the data set description. Dimension reordering methods for SPLOM based on the similarity between projections have been proposed [WAG05], but no *quality-aware sorting* methods have been presented so far. This motivates us to adopt a *quality-aware sorting* concept and to start investigating the advantages of such an approach. Note that the concept of dimension reordering can be applied to any matrix-based visualization. In Section, 4.4 we also apply a dimension reordering for our PCM, but for the ease of understanding we use SPLOMs in this section. The pipeline of our framework for quality-aware reordered SPLOMs

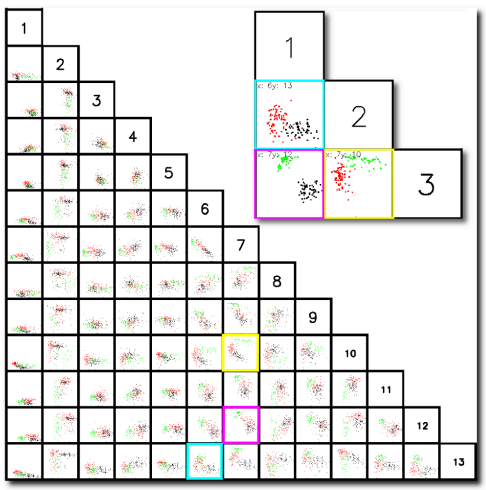


Figure 4.5: Results of the C-SPLOM for the Wine data set. Visualizations with only little overlap are preferred, so the difference between the wine cultivars becomes clearer. The best visualizations for each class pair are shown in the C-SPLOM.

(D-SPLOM) is shown in Figure 4.6 and explained in the following. As a pre-process to reordering, we initially apply a quality metric $Q(s_{(a,b)})$ to each scatterplot $s_{(a,b)}$. This quality metric ought to be a scalar one so that it rates the scatterplot unambiguously with a single number. Apart from that, it can be any useful metric [WAG05, SNLH09], including the metrics presented in Chapter 2. Furthermore, we need this quality metric to estimate the quality of each dimension itself. Once we have $n-1$ scatterplots for each dimension in an n-dimensional data set, we consider $n-1$ quality metrics (one per plot) to compute the overall quality metric for the respective dimension.

For each dimension d, we compute a dimension-metric as the base for reordering. A dimension-metric Q_d is a scalar function $Q_d : \mathbb{R}^{n-1} \to \mathbb{R}$ over all quality metrics $Q(s_{(a,b)})$ of a dimension d: $D_d = Q(s_{(d,i)})$, where $i \neq d$ and $i \in [1,\ldots,n]$. It combines the quality value over all scatterplots which contains the dimension d. There are exactly n dimension-metrics for an n-dimensional data set. Different functions can be

used for computing D_d as long as it is guaranteed that the metric values are comparable to each other, such as the *mean*, a *PCA*, or the *variance* over the pre-computed quality metrics. We use the sum over all quality metrics as dimension-metric for this experiment:

$$D_d = \sum_{i=1,\ i\neq d}^{n} Q_{(s_{(d,i)})}. \tag{4.2}$$

This metric produces the same partial order between the dimension-metrics as the mean, but has the advantage that it is easier and faster to compute. In this last step, we make use of the computed information to reorder an $n \times n$ SPLOM. Because such a SPLOM is symmetric, we use the upper triangular matrix for display. First, we allocate to each dimension its quality metric value D_d. We sort all quality-metric/dimension pairs (D_d, d) by means of a simple partial order $(\geq)$ with respect to the quality metric D_d, which gives us a dimension-ranking $r = (sort\{(D_d, d)\}; \geq)$. The dimension $r[0].d$ from ranking r describes the *best* dimension, and $r[n].d$ the *worst* one.

We map a dimension d to its position in the ranking r and, depending on that mapping, we reorder the scatterplots in the SPLOM and get the quality-based reordered SPLOM (D-SPLOM), as can be seen in Figure 4.6.

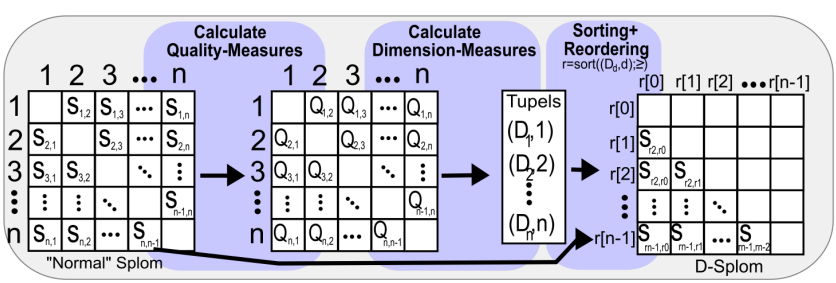


Figure 4.6: Overview of the dimension reordering process.

4.6.1 Experiments

To evaluate our concept, we test it on real classified and non-classified multidimensional data sets. For classified data, we apply the *Class Density Measure* (CDM) (Chapter 2) as a quality metric to the *Olives* data set [ZNLG94], cf. Section 4.5.1. The CDM assigns higher values to scatterplots with a better separation between the classes. The result of the reordering is shown in Figure 4.7. For non-classified data, we apply the *Rotating Variance Measure* (RVM) (Chapter 2) as quality metric to the Parkinson data set. This data set has 13 dimensions, no classes, and 197 samples. The RVM rates the linear and non-linear correlation within the scatterplots with respect to its two dimensions. The result is shown in Figure 4.8.

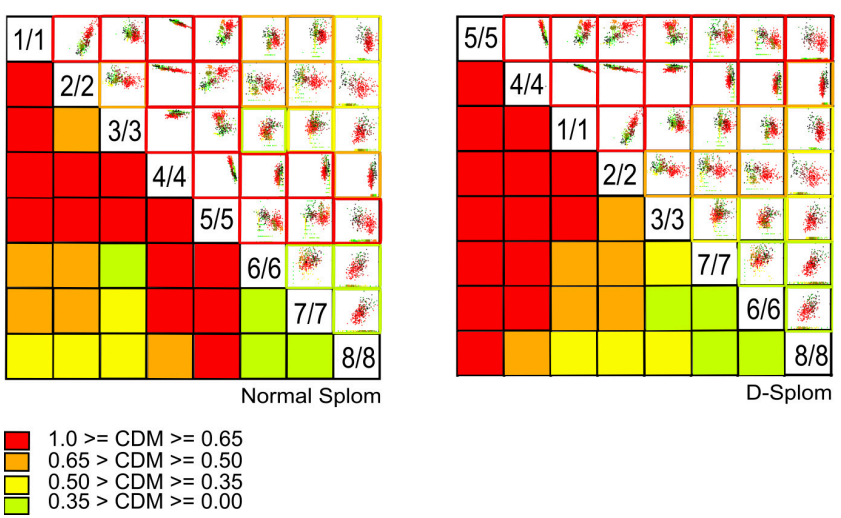


Figure 4.7: Evaluation of our D-SPLOM on the class-based *Olive oil* data set using the CDM: Ordinary SPLOM (left), the resulting D-SPLOM (right)

In Figure 4.7 and 4.8, relevant scatterplots are colored more red than non-relevant. It is easy to see that before reordering both types of scatterplots are distributed over the whole SPLOM. After reordering, relevant and non-relevant scatterplots in the D-

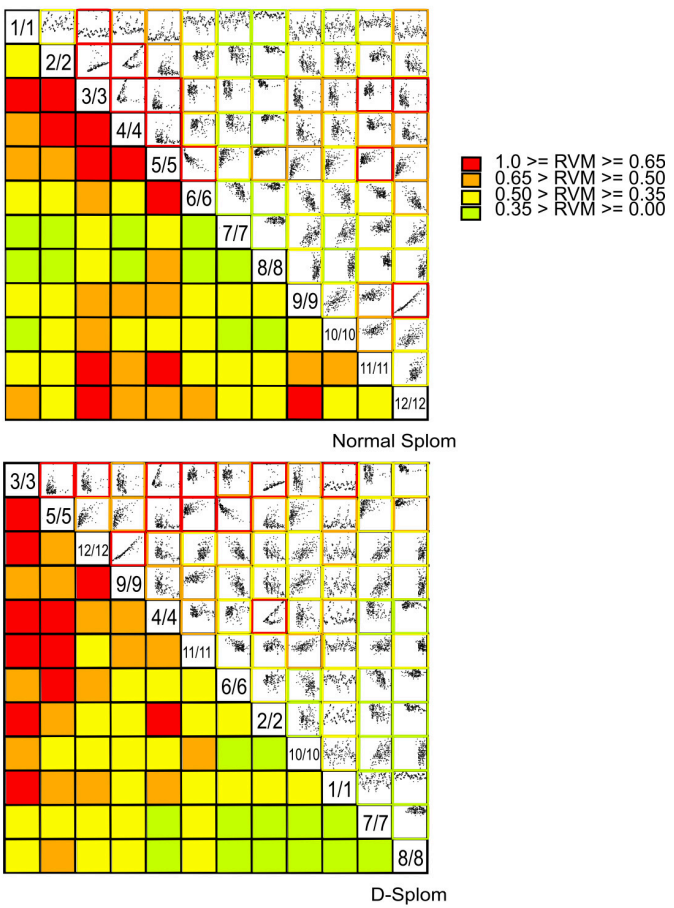


Figure 4.8: Evaluation of our D-SPLOM on the non-class-based *Parkinson* data set: Ordinary SPLOM (top), the resulting D-SPLOM (bottom).

SPLOM are mostly separated from each other. We can observe that the quality-aware reordering reduces the region of interest, speeding up the visual search. Quality-aware reordering has practical advantages and enhances the visual quality of SPLOMs. Depending on the data set, however, some dimensions might contain outliers. This may

happen if the quality metric used assigns a low value to most visualizations of one dimension, but a high value to only a few, as for example, dimension 4 in the SPLOM shown in Figure 4.8. Our color coding allows for easy recognition of such plots. In the future, it is worth investigating how far the quality-aware framework is capable of suppressing non-relevant scatterplot from SPLOMs, speeding up even more the visual search task.

4.7 Data Normalization

Any visualization technique representing data values by color requires to map the data values d_i of a data set D onto a specific given color map. One can think of a color map as a lookup table that assigns data values in the range $[0,1]$ to specific colors. In order to accomplish this, one needs to compute the transformed values p_i in the range $[0,1]$ by a transformation T, i.e. $p_i = T(d_i)$. These values are then directly used to query the color map. The goal of data normalization is therefore to derive a suitable transformation T that results in a color mapping which reveals the interesting structures in the data and allows for easy comparison and correlation analysis in the final visualization. There exists a great variety of possible data transformations, ranging from a simple addition of a constant to multiplying, squaring, raising to a power, converting to logarithmic scales, inverting, reflecting, taking the square root, histogram equalization and many more [HD79, Mic89].

As the term *interesting structures* is not well-defined, these methods usually aim at either improving the *normality*, i.e. normal distribution, or simply exploiting the whole color range, e.g. by histogram equalization. The term *normal distribution* refers to a particular way in which observations of a variable tend to pile up around a particular value rather than being spread evenly across a range of values. Many visualization methods assume the premise of a normal distribution of the data [Mic89]. Therefore, examining and understanding the data is a necessary step to decide when and which function shall be used to normalize it. There are different forms to verify the normality of the distribution, from observing its frequency distribution histogram to

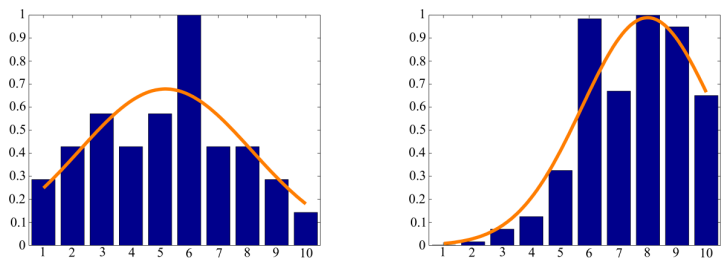


Figure 4.9: Left: Example of a 'well-shaped' distribution histogram. Right: Skewed histogram where most of the data values are large, making it difficult to find a good data transformation.

more sophisticated normality tests such as Kolmogorov-Smirnov, Anderson-Darling, Cramér-von-Mises and Lilliefors tests [GSF77]. Figure 4.9 shows on the left the distribution histogram of a data set that has approximately a normal distribution and on the right a data set where the distribution is skewed to the right.

Histogram equalization aims at transforming the distribution such that the cumulative histogram becomes a straight line [GW06]. More space in the histogram is reserved for values of high occurrence, and the whole range of possible values is exploited. This is helpful for discrepancy analysis, but, unfortunately it can happen in standard histogram equalization algorithms that equal data values are mapped to different colors due to the inflexible binning. In addition, outlier analysis becomes more difficult, and any visual connection to the absolute values is completely lost.

4.7.1 Data Driven Color Mapping

As shown in the previous section there exist many different transformation techniques to normalize the distribution of a data set. However, from a perceptual point of view, normality of the data is not the ultimate goal for visual analysis. Imagine a simple two-peak distribution. There is no single transformation that is able to create the bell-shaped normal distribution while preserving the discrimination of the two obvious

clusters. From a perceptual point of view, we state several goals that a transformation should aim for:

1. if $d_i = d_j$ then $p_i = p_j$ (preserve equality)
2. if $d_i \leq d_j$ then $p_i \leq p_j$ (preserve ordering)
3. if $(0 < |d_i - d_j| < \tau_1),\ \tau_1 > 0$ then $|p_i - p_j| > \tau_1$ (increase discrimination between similar values)
4. if $(|d_i - d_j| > \tau_2)$, then $\tau_1 < |p_i - p_j| < \tau_2$ (increase similarity if values differ largely to save space in the mapping domain, but preserve discrimination)

with d_i being the data values of a data set D and p_i the respective transformed values. While the first two statements appear rather trivial, the third and fourth are critical as they apparently claim opposing goals. While the third tries to increase the gap between different values to support the discrimination of similar values, the fourth goal tries to decrease the gap between values for a better exploitation of a given mapping range. In the following we describe a solution which is well suited for various data distributions.

Data projection Our algorithm proceeds as follows (Figure 4.10). We first sort our data values in ascending order. Each data value d_i in this sorted array can now be thought of as a two-dimensional vector $\mathbf{v_i}$ whose x-component is its position on the x-axis, i.e., its position in the sorted array, and whose y-component is the data value itself. To meet our goals, we now project each vector onto the diagonal which is spanned by the smallest ($\mathbf{v_1}$) and largest element ($\mathbf{v_n}$) and normalize the value to lie in the range $[0,1]$:

$$\text{Let } \mathbf{v_{diag}} = \mathbf{v_n} - \mathbf{v_1}, \text{ and } p_i = \frac{\left\langle \mathbf{v_i}, \mathbf{v_{diag}} \right\rangle}{|\mathbf{v_{diag}}|^2}, \tag{4.3}$$

where $\langle , \rangle$ depicts the dot-product. This first step is similar to a histogram equalization, but with the advantage that very different values will never be assigned to the same projected value, as can happen in standard histogram equalization (though they can

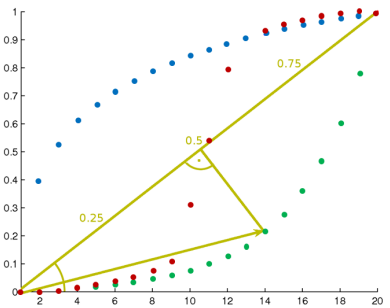


Figure 4.10: The three dotted lines depict three different sorted data sets, resulting in a logarithmic (blue), an exponential (green) and an s-shaped curve (red). Interpolating the distance of the data points in the x-direction also changes the angle α. Choosing a low value for α will treat all data points equal, while a higher value will emphasize outliers.

still be mapped to the same color if the color map contains too few entries). This can be a valid solution for our goals two, three and four: The ordering is preserved, and the projection has a spreading effect for very similar values along the diagonal but reduces larger differences. However, equal values can still be mapped to different colors. To remove the violation of the equality statement, we search for equal values d_i and their projected values p_i and map them to the mean of the projected values, i.e.

$$p_i = \frac{1}{w}\sum_{j=1}^{N}\delta(i,j)p_j \text{ ,where } w = \sum_{j=1}^{N}\delta(i,j) \text{ ,}$$
$$\text{and } \delta(i,j) = \begin{cases} 1 \text{ if } d_j = d_i \\ 0 \text{ otherwise} \end{cases} \tag{4.4}$$

Interpolation After data transformation, interpretation of the data values can be difficult for the user, as the visual link to the absolute values has been lost. In addition, a good data transformation necessarily transforms largely differing values to more similar values in order to better exploit the color map range. As it is essentially not possible to achieve both goals simultaneously, namely interpretability of absolute values and a good distribution in the color map range, we allow the user to interactively change the

projection described in Sect. 4.7.1. We do this by providing a slider to interactively change the angle of the diagonal $\mathbf{v_{diag}}$. To assure that $\mathbf{v_{diag}}$ still starts at v_1 and ends at v_n, we spread the x-values of each vector accordingly, so that the distance d_x of the x-value between v_1 and v_n is

$$d_x = \frac{y_{max} - y_{min}}{\tan(\alpha)} , \tag{4.5}$$

where α is the user specified angle, see Figure 4.10. Choosing a high value for α emphasizes outlier values, as it is close to linear scaling (choosing $\alpha = 90^\circ$ corresponds to linear scaling). A very small value for α is similar to histogram equalization, which is beneficial if comparisons between specific values are important. This smooth transition is easier to use and makes the visualization more comprehensible than simply testing arbitrary data transformations.

4.7.2 Experiments

We test our approach on several real-world and synthetic data sets, Figure 4.11 shows an example of a choropleth map of the U.S.A. for the county-level unemployment rates of 2009 from the Bureau of Labor Statistics using only six different colors. In the first map, a linear color mapping is chosen. Outliers can be easily detected, but discrimination of more similar values is difficult. The second map uses logarithmic color mapping. The result looks better but is still not pleasing, as, e.g., large portions of the whole state of Missouri or New York are mapped to the same color. Using a low α value, our result on the right shows a contrasty good distribution of color values, which eases comparison or similarity analysis. In Figure 4.12 the usefulness of our transformation approach is depicted for the same data set and color map. Almost linear scaling on the left is helpful if absolute values are of interest or outliers are to be detected, as, e.g., the high unemployment rate of 30.1% in Imperial County, California. Decreasing the angle for our data projection technique increases discriminating properties between the counties with more similar values, making it possible to see the more subtle differences, e.g. in Alaska between Matanuska-Susitna with 8.8%, Valdez-Cordova with 6.4% and South-East Fairbanks with 8.1%, see Figure 4.13.

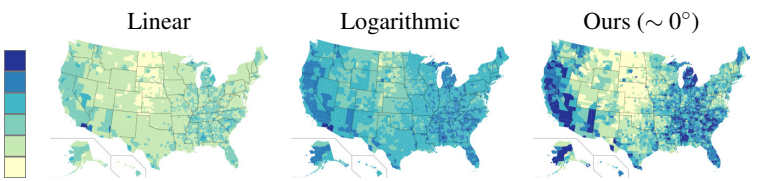


Figure 4.11: Choropleth Map of the U.S.A. displaying county-level unemployment rates. Left: The color map used for visualization. Middle left: Linear scaling of the data values does not reveal any significant information except for two outliers on the California-Arizona border. Middle right: Logarithmic scaling (base 10) of the data values does not significantly improve readability. Right: Our method shows a clear discrimination between the different counties and facilitates similarity analysis.

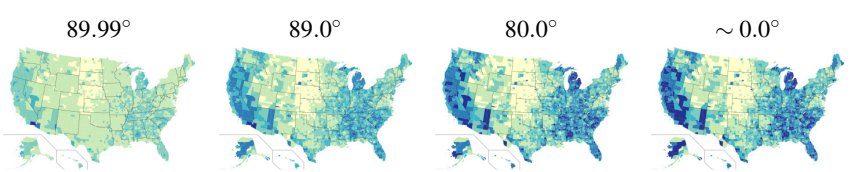


Figure 4.12: Example of our interpolation technique. A high α value resembles an almost linear scaling (left). Outliers can be easily detected, but the finer differences between the counties can get lost, e.g. in New Mexico, Wyoming, Texas, or Alaska. A low α value results in better discrimination properties, and the subtle differences between the counties become visible (right). However the information about the absolute scale gets lost. By providing intermediate values by interpolating between these two extrema, the user can get both, a better discrimination, a visual link to the absolute value, and easy outlier detection. The angular interpolation value α is shown at the top of each image.

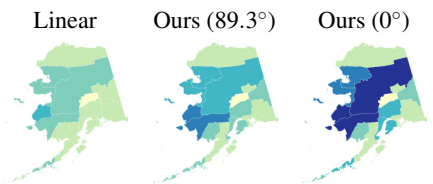


Figure 4.13: Close-up of Alaska from Figure 4.12. The left image uses linear scaling, the right image uses our projection technique, and the image in the middle uses our interpolation technique with an angular value of 89.3. Subtle differences become obvious with our technique.

We also apply our method to jigsaw maps [Wat05], which are a 2D space-filling visualization. Exemplarily we apply our color mapping technique to the *Ozone Level Detection* data set [ZF08] with 2536 instances and 73 dimensions. The samples of the data set were collected in a eight hour peak from 1998 to 2004 at the Houston, Galveston and Brazoria area. In Figure 4.14 we visualize the 35th dimension (T8), which depicts the temperature at 8 a.m. in the morning throughout several years. Linear scaling, on the left of Figure 4.14, results in a rather dull appearance. Using our approach the contrast is increased, making the difference between different seasons more obvious.

4.8 Discussion

In this chapter, we have presented two new visualization matrices to support the visual analysis of high-dimensional data sets: A class-based scatterplot matrix for data sets with label information that supports the analysis of pairwise relationship between classes, and a parallel coordinates matrix that allows examining correlations between all possible dimensions using parallel coordinates plots. We also proposed an information-bearing reordering framework that can improve the visual analysis task for any matrix-based visualization method, and we have shown that our quality-based visualization matrices, together with the presented reordering framework, successfully reduces the

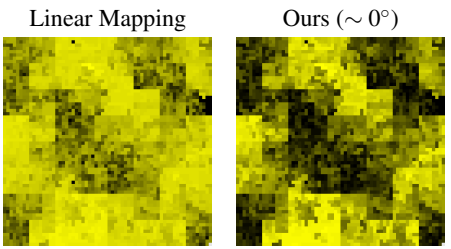


Figure 4.14: Left: linear color mapping for a jigsaw map [Wat05] of the 35th dimension (T8) of the Ozone data set [ZF08] results in a rather dull appearance. Right: our method is able to create a contrast-rich result.

region of interest in visualization matrices. The investigation of more sophisticated ranking functions for dimension reordering is still an open issue and needs more investigation. Additionally, we have presented a simple yet effective method to increase the readability of pixel-based visualizations: a new automatic data transformation which preserves important aspects of the data distribution, needed for analysis. Our method allows for a better utilization of the whole color range by increasing discriminating properties between similar values while reducing the discrepancy between different values. Our algorithm spares the user the time-consuming task of testing different, unrelated transformations and potentially speeds up the visual exploration of the data. In addition, our interpolation technique enables the user to also keep the visual link to the absolute values of the data set.

Chapter 5

Data Generation

5.1 Motivation

Generating synthetic data sets for testing is a common practice in many research areas. In situations where large amounts of real data may be difficult to obtain, due to budget, time or privacy concerns, generating synthetic data poses a viable alternative. As a substitute for real data, synthetic data can be used to provide a controlled testing environment that meets specific conditions. This is especially important in verification, simulation, and proof of concept studies.

There are numerous algorithms and tools that can be used to create synthetic data. However, most of the work done so far was developed for specific applications, for example, synthetic data generation to test Information Discovery and Analysis Systems [JSL+05, JLR+06] or software testing [MMSW97].

The common solution to generate synthetic data is to use well-known statistical tools or to develop small problem-oriented applications; either can be a time-consuming task. Addressing this shortcoming, we propose a framework for synthetic data generation that allows creating multivariate data sets based on visual specifications. Our approach was originally developed to create test cases for automated visual exploration algorithms, i.e. the methods presented in Chapter 2, 3 and 4, but it can be directly applied to any application that works on multivariate data sets, such as

learning and data mining algorithms. Such visual exploration algorithms are commonly used to search in high-dimensional data sets for hidden structures that can be observed in lower-dimensional projections of the data (two or three dimensions). A typical, synthetically generated data set to test such algorithms must have a large number of dimensions. The data values in most dimensions are randomly generated, and multidimensional structures are then defined in the remaining dimensions. Some examples of such structures are correlations or clusters between selected dimensions. A good data set does not necessarily exhibit only features that can be found by any visual analysis tool, but it may contain complex structures that are difficult to find, e.g., non-orthogonal structures, or non-linear correlation between dimensions. Using synthetic data sets, it can be reliable verified if all defined structures can be found by some algorithm.

We propose an approach to generate different patterns in the data, including multidimensional clusters, correlations, and other trends. The points of the data set are created by sampling statistical distributions in a multidimensional space: First, the user gives basic information about the desired data set, e.g. the number of dimensions and number of samples. Second, the user defines a set of structures in the multidimensional space that are represented by *probability density functions*. Finally, data points are generated by sampling these functions. Our framework allows the user to create classified as well as unclassified high-dimensional data sets. Complex structures are inserted using painting tools. This way, correlations between attributes and clusters of different forms and dimensions can be simulated. Additionally, the algorithm is able to create structures based on an analysis of existing real data.

5.2 Related Work

Generating synthetic data is an important tool that is used in a variety of areas, including software testing, machine learning, and privacy protection.

In software testing, synthetically generated inputs are used to test complex program features and to find system faults. The difficult and expensive process of generating

test data that meets specific fault-based testing criteria has traditionally been done by hand. In 1991, DeMilli and Offutt claim to have made the first attempt to automatically generate such test data [DO91], introducing the approximation technique of *constraint-based testing*. More recent research suggests the use of genetic algorithms to generate adequate test data [MMSW97, PHP99]. To this day, generating test data remains an active research topic.

Another application area is the field of machine learning, where synthetic data can be used to balance training data. For example, when training a support vector machine to distinguish between a number of different patterns, the amount of training samples for each pattern should be equal. In this field, it is a common problem that over-representing a certain pattern can introduce artificial bias towards that pattern. In order to prevent this problem, training data sets may be equalized by generating additional synthetic training data [GV04].

Synthetic data is also commonly used in scenarios where collecting real data is not an option, due to budget, time or privacy concerns. Synthetic data sets can be designed to obscure sensitive information while still maintaining the statistical properties of the original data [Rei02]. This is exploited for testing the reliability of data-mining tools that have to operate on complex multivariate data. For the simulation of real, high-dimensional data, complex statistical interdependencies within the multivariate data set can be modeled using semantic graphs [JSL+05, JLR+06].

Another approach is to generate synthetic data sets using the programming language R [Dal02]. This statistical computing environment offers a strong, script-based approach using a variety of readily available statistical distributions.

Our framework is heavily focused on generating synthetic data with very specific *visual* properties. Many of the classical statistical generation techniques do not apply to our case since we need synthetic data that mimics visual trends of real data sets. The motivation to develop this framework was the evaluation of automated visual exploration algorithms that support the analysis of high-dimensional data. Diverse automated visualization techniques [PWR04] and quality metrics [TT85, WAG05] have been designed to analyze these high-dimensional data sets, including the methods

presented in the previous chapters. They provide the user with appropriate projections, displaying information deemed useful according to user-defined constraints. Our framework is designed to generate synthetic data sets that can be used to test and evaluate such visual exploration techniques.

5.3 Overview

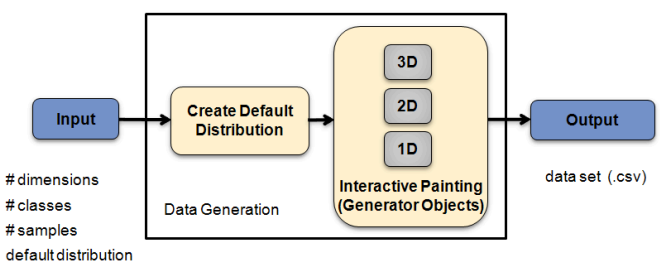


Figure 5.1: Framework of our synthetic data set generator.

Figure 5.1 depicts the data creation pipeline. These are the steps that our generation algorithm goes through when creating an n-dimensional data set: First, some properties of the data set are defined as input. These properties include: the number of dimensions, number of data samples, classes, a default distribution, and the data type of each dimension. According to these parameters, a default n-dimensional probability density function (PDF) is created that is used to generate a default data set. Each data sample is assigned a random position in this space. In the second step, structure may be added to chosen dimensions. The user can adjust and model desired structures using generator objects. Our framework has three kinds of such objects: A one-dimensional PDF that allows manipulating each dimension separately, a two-dimensional PDF to manipulate orthogonal projections of two dimensions simultaneously, and a three dimensional generator object that is defined using a special plane, henceforth termed *probability distribution plane* (PDP). In the last step, user-controlled random noise may be additionally applied to all dimensions to simulate irregularities found in real

data. All properties and generator objects are stored in an XML file to allow for future editing if necessary. The generated sample points are exported in a comma-separated values (*.csv*) file format.

5.4 Data Generation

Synthetic data generation algorithms enable users to create artificial data sets with desired characteristics. In our framework, the user defines these characteristics as part of a *generator*. We define a generator as the central structure that contains all necessary information to generate an n-dimensional data set. This includes the number of dimensions and the amount of samples that are to be generated. Furthermore, the distribution of these samples is defined by a set of *generator objects*. Each generator object is a small module representing the relationship between different dimensions. The user may attach multiple objects to any of the n dimensions of the generator. These objects taken together define the distributions in our n-dimensional data set.

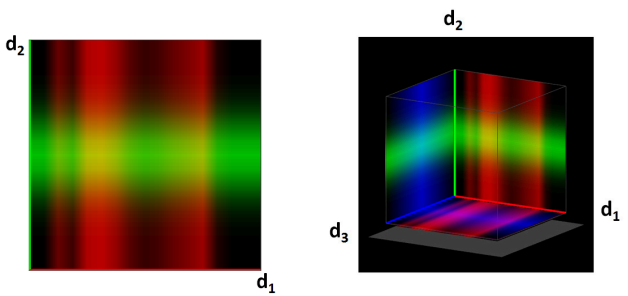
While the list of available generator objects is designed to be easily extended, we use three specific modules that can model relationships between one, two, or three dimensions, respectively.

5.4.1 One-Dimensional Probability Density Functions

The first generator object is a *one-dimensional probability density function* (one-dimensional PDF). A PDF $\rho(x)$ models the distribution of a variable x along a single dimension [Bul79]. $\rho(x)$ describes the likelihood that a random variable will fall on $x \in \mathbb{R}$. Any number of one-dimensional PDFs ρ_i can be utilized to define a multidimensional PDF in such a way that:

$$\rho(x_1,\ldots,x_n) = \prod_{i=1}^{n} \rho_i(x_i).$$

Figure 5.2 shows an example for two and three dimensions. In our framework, we use n of these PDFs to define our n-dimensional probability density space.



(a) $n = 2$ (b) $n = 3$

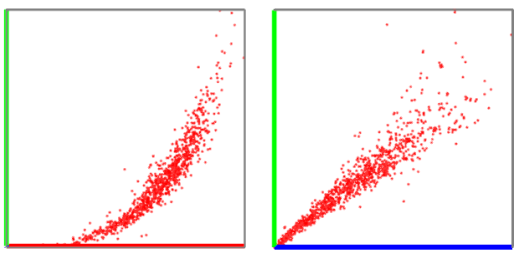
Figure 5.2: Generator schematics, showing one-dimensional PDFs ρ_i (marked red, green and blue) along the axes of the probability density space in (a) two and (b) three dimensions. This view illustrates how a set of one-dimensional functions span a multi-dimensional PDF, with color intensities representing the respective probability distributions.

5.4.2 Two-Dimensional Probability Density Functions

The second module is a *two-dimensional probability density function* (two-dimensional PDF). This module is analogous to the previously described *probability density function*, and is used to define conditional relationships between two distinct dimensions of the n-dimensional probability density space. Mapping multidimensional data sets in lower-dimensional projections is a common practice in information visualization. Particularly, two-dimensional projections are extensively used in diverse visualization tools, e.g. XmdvTool [War94] and GGobi [STBC03]. Most commonly, such tools show two dimensions at a time, using an axis-aligned projection of the data. Appling this approach to data generation, a two-dimensional PDF allows the user to design features from a familiar perspective.

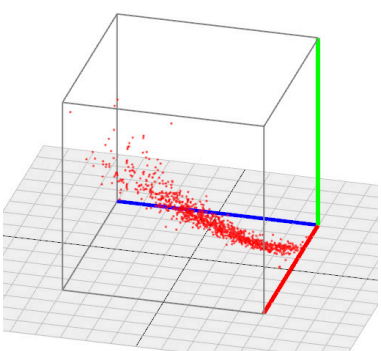
Beyond the ability to generate axis-aligned structures, a set of these PDFs may be combined to define more complex structures across an arbitrary number of dimensions, as can be seen in Figure 5.3. The data set seen in (a) and (b) was generated using two independent, two-dimensional PDFs attached to the green and red, or green and blue

dimensions, respectively. Note how the combination of these two PDFs generates a structure in three-dimensions that satisfies both projected distributions.



(a) Axis-aligned projection (green and red dimensions)

(b) Axis-aligned projection (green and blue dimensions)



(c) Axis-independent projection

Figure 5.3: A data set consisting of 1000 samples. It was generated using two different two-dimensional PDFs defined between (a) the green and red, and (b) the green and blue dimensions, respectively. (c) is an axis-independent view that shows the three-dimensional structure that has been created by combining the two PDFs.

5.4.3 Probability Distribution Planes

In addition to axis-aligned distributions, the user may also demand data sets containing asymmetrical clusters and correlations that cannot be represented by one- or two-

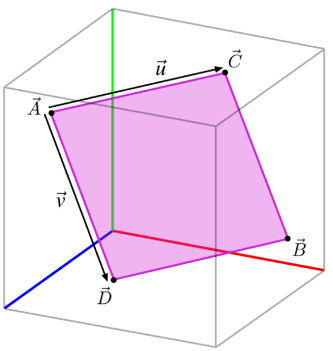


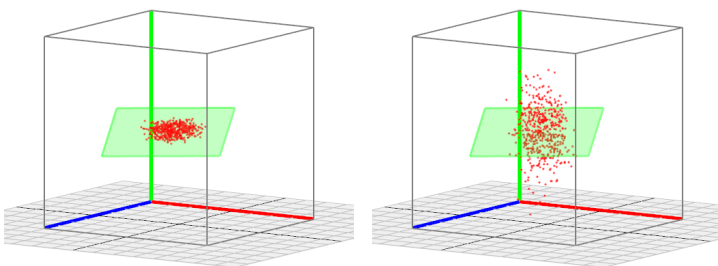
Figure 5.4: Probability Distribution Plane (PDP). $\vec{A}$, $\vec{B}$, $\vec{C}$, and $\vec{D}$ define the corner points of the PDP generator object. A two-dimensional PDF $\rho(\|\vec{u}\|, \|\vec{v}\|)$ is mapped onto the quadrilateral marked in purple.

dimensional axis-aligned PDFs. For that purpose, we introduce our final generator object: The *probability distribution plane* (PDP).

A PDP is defined on a three-dimensional subset of the n-dimensional probability density space. It is described by four points $\vec{A}, \vec{B}, \vec{C}, \vec{D} \in \mathbb{R}^3$, so that all four points lie on a common plane, Figure 5.4. These four points are the corners of a quadrilateral, onto which a two-dimensional PDF $\rho(\|\vec{u}\|, \|\vec{v}\|)$ is mapped. This allows the user to create a non-orthogonal two-dimensional PDF across three different dimensions.

In addition, we developed a scattering tool that allows the user to extend the two-dimensionally defined distribution into the third dimension by using a noise-like function that is applied along the plane normal. This function is controlled by two parameters: Scattering type and scattering intensity. The scattering type describes the behavior of the noise that is being applied along the plane normal during data generation. Possible types include Gaussian, linear, constant, and density scattering (which depends on the local density $\rho(\|\vec{u}\|, \|\vec{v}\|)$.

Figure 5.5 shows an example of how scattering affects data points generated from a PDP. (a) depicts the points generated without any scattering applied to them, while (b) portrays the points scattered along the normal of the plane using *Gaussian scattering*.



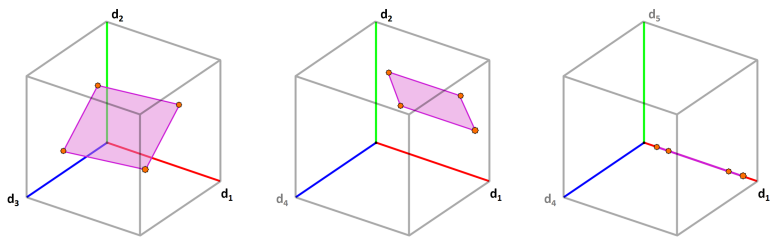
(a) No scattering (b) Gaussian scattering

Figure 5.5: A data set of 500 points generated using a PDP

The PDP is designed to easily create structures within the human visible domain. It allows the user to model axis-independent clusters, correlations, and other features in an intuitive way. As can be seen in Figure 5.6, PDPs not only influence the exact subset of dimensions they are defined in, but they also influence all other subsets of the n-dimensional data set that contain any of the dimensions associated with the PDP. This property can be utilized to create visual and statistical patterns beyond the three dimensions of the generator object. PDPs may be combined with other generator objects in order to shape more complex structures across numerous dimensions.

5.4.4 Generation Algorithm

Once the user has finished setting up generator objects, the final data set can be generated. During this process all of the provided generator objects $g \in G$ are taken into account in order to generate an n-dimensional data set consisting of a total of m data samples $\vec{s_j}$ so that:



(a) (b) (c)

Figure 5.6: (a) A PDP that is defined on a three-dimensional subset of the n-dimensional probability density space. The PDP influences all subsets of dimensions connected to the three dimensions it is associated with. (b) The visualization of the defined PDP after switching one of the dimensions in the visible three-dimensional subset. (c) The visualization of the defined PDP after switching two of the dimensions in the visible three-dimensional subset

$$\vec{s_j} := \begin{pmatrix} \tau_{j,1} \\ \tau_{j,2} \\ \vdots \\ \tau_{j,n} \end{pmatrix} + \begin{pmatrix} v_{j,1} \\ v_{j,2} \\ \vdots \\ v_{j,n} \end{pmatrix}, \quad \forall j = 1,2,\ldots,m$$

where $\vec{\tau_j}$ is the original value of the j^{th} data sample and $\vec{v_j}$ is a vector of trivial noise values that may be of a Gaussian, linear, or constant distribution.

Algorithm 1 shows a simplified version of our generation algorithm. We start by generating each sample $\vec{\tau_j}$ of the data set. At the beginning, all attributes of the sample $\vec{\tau}_{j,i}$ are undefined and will be computed from the generator objects. Every dimension is assigned exactly one one-dimensional PDF, but may also have an arbitrary number of two-dimensional PDFs and PDPs associated with it. To determine $\vec{\tau}_{j,i}$ for each dimension, we randomly choose a generator object g from our defined set G. Since there may be more than one PDF defined per dimension, these PDFs may contradict each other in some ranges of the dimension. Random selection allows us to average these contradicting PDFs. Depending on the kind of the selected generator object, the value of the attribute is computed as follows:

Algorithm 1 Generation algorithm.

for $j = 0 \rightarrow m$ **do**
  for $i = 0 \rightarrow n$ **do**
    if $\tau_{j,i}$ is undefined **then**
      $g \leftarrow g_{rand} \in G$ {(1) Select a random generator object}
      if g is PDP **then**
        if τ_{j,dim_k} is undefined, $\forall dim_k \in g$ **then**
          $\tau_{j,dim_k} \leftarrow$ rand(g), $\forall dim_k \in g$
        else
          go to (1)
        end if
      end if
      if g is two-dimensional PDF **then**
        if τ_{j,dim_k} is undefined, $\forall dim_k \in g$ **then**
          $\tau_{j,dim_k} \leftarrow$ rand(g), $\forall dim_k \in g$
        else
          if τ_{j,dim_1} is undefined, $\{dim_1, dim_2\} \in g, dim_1 \neq dim_2$ **then**
            $\tau_{j,dim_1} \leftarrow$ rand(g, τ_{j,dim_2})
          end if
        end if
      end if
      if g is one-dimensional PDF **then**
        $\tau_{j,dim} \leftarrow$ rand(g), $dim \in g$
      end if
    end if
  end for
end for

- If g is a PDP, we verify whether all associated dimensions $dim_k \in g, k = 1,2,3$ are still undefined for this sample $\vec{\tau}_j$. If this is the case, then the PDP is used to generate values, not only for the i^{th} dimension, but for all three dimensions involved. Otherwise, this particular PDP object is discarded for now and another generator object g is selected instead.

- If g is a two-dimensional PDF, we first verify the two associated attributes. If neither value of the two involved dimensions $dim_k \in g, k = 1,2$ is defined, we can safely generate new values for each of the attributes. However, if one of these values has already been defined in a previous step, we generate only the value of the undefined attribute using this generator object, constrained by the previously generated value.

- One-dimensional generator objects are only selected when no PDP or two-dimensional PDF can be used to generate the values. Whenever there are no other valid generator objects defined on the current dimension, the one-dimensional PDF is selected instead. A single independent value for the attribute on dimension $dim \in g$ is then generated based on this PDF.

Once all attributes of $\vec{\tau}_j$ have been processed, the sample is fully defined. This process is repeated until all samples are generated. This execution order has a bias towards complex generator objects since PDPs and two-dimensional objects are preferred.

To better illustrate our approach, a brief step-by-step example is given in Figure 5.7. It demonstrates how a data sample is generated using our algorithm.

In this example the data sample begins completely undefined in step 0. In step 1, a three-dimensional generator object (PDP) is randomly selected. Three attributes $\tau_{j,1}, \tau_{j,4}, \tau_{j,n-2}$ are generated at the same time and from the same distribution function. In step 2, the one-dimensional generator object (one-dimensional PDF) is randomly selected. Therefore, only the current attribute $\tau_{j,2}$ is generated. In step 3, the algorithm selects a two-dimensional generator object (two-dimensional PDF). In this case, the generator object is associated with two attributes, one of which has already been

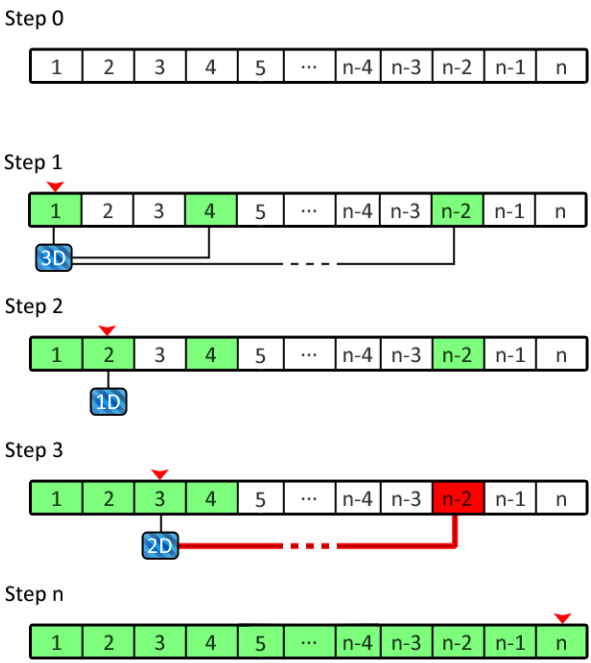


Figure 5.7: Example of how an individual data sample $\vec{\tau}_j$ is generated. The numbered cells represent the attributes $\{\tau_{j,1},\ldots,\tau_{j,n}\}$. Colouring indicates whether an attribute is defined (green) or not (white). Below the cells is the currently selected generator object, with lines connecting it to all associated dimensions. Red colouring indicates an association with a previously already defined attribute.

defined. Thus, only the other attribute at $\tau_{j,3}$ is generated, but constrained by the value of the attribute at $\tau_{j,n-2}$. This process continues until the data sample is completely defined in Step n. Note that given the nature of the algorithm, it is likely that $\vec{\tau}_j$ is fully defined even before the n^{th} step, in which case it is safe to proceed to the next sample point $\vec{\tau}_{j+1}$.

Our data samples are eventually saved as comma-separated values (*.csv*), in a way that represents an $m \times (n+1)$ matrix with the values at $n+1$ being the class ID of the sample, so that:

$$\begin{pmatrix} \vec{s_0}^T \\ \vec{s_1}^T \\ \vdots \\ \vec{s_m}^T \end{pmatrix} = \begin{pmatrix} s_{0,0} & s_{0,1} & \cdots & s_{0,n} & s_{0,class} \\ s_{1,0} & s_{1,1} & \cdots & s_{1,n} & s_{1,class} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ s_{m,0} & s_{m,1} & \cdots & s_{m,n} & s_{m,class} \end{pmatrix}.$$

5.5 Sampling a Probability Density Function

Using our framework, the user may define an arbitrary one- or two-dimensional *probability density function* as input by simply drawing it on the screen. This input is then interpreted as a finite list of values representing a PDF. Directly interpolating these values into a continuous function and using common statistical methods to sample this distribution, however, can be computationally expensive depending on the given function. For that reason, we develop an alternative algorithm that allows us to quickly sample any function the user may define as input.

5.5.1 Input Functions

Before we can assume that the finite list of discrete values that the user provides as input represents an actual *probability density function* ρ, we have to prove that the input, after interpolation, will adhere to the definition of a PDF.

First, the user interface is designed in such a way as to only allow values greater than or equal to 0, thus fulfilling:

$$\rho(x) \geq 0, \quad x \in \mathbb{R}. \tag{5.1}$$

Second, the input is limited to a user-defined range, given by a lower bound α and an upper bound ω. Our system further assumes that all values beyond that range are 0. This yields:

$$\lim_{x \to -\infty} \rho(x) = \lim_{x \to \infty} \rho(x) = 0. \tag{5.2}$$

Finally, a user-drawn function is considered to be implicitly normalized:

$$\int_{-\infty}^{\infty} \rho(x) dx = 1. \tag{5.3}$$

(5.1), (5.2) and (5.3) together define a continuous PDF [BSMM01, p.772].

We therefore know that the given input, after interpolation, adheres to the mathematical definition of a continuous PDF.

5.5.2 Sampling a Discrete Function

When generating random samples from a continuous PDF ρ, we make use of ρ's *cumulative probability function P*. A *cumulative probability function* [Bul79, pp.36] is defined as

$$P(t) := \int_{-\infty}^{t} \rho(x)dx, \ \ \forall\, t,x \in \mathbb{R}.$$

Since, in our case, the continuous function ρ is represented by a list of l discrete values ρ_{arr}, we can calculate $P_{arr}[x]$ as

$$P_{arr}[x] := \sum_{i=0}^{x} p_{arr}[i], \ \forall x \in [0,\ l-1] \cap \mathbb{N}.$$

Generating a random variable r with distribution ρ_{arr} is equivalent to selecting an array-index $i \in [0, l-1]$ and mapping that index onto $[\alpha, \omega]$:

$$r := (\alpha + i(\omega - \alpha)),$$

being α and ω the lower and upper bounds, respectively.

Since most trivial random number generators can only generate uniformly distributed random variables, we cannot directly reproduce the arbitrary distribution described in ρ_{arr}. Therefore, we have to map these uniformly distributed random variables onto a weighted representation of ρ_{arr}. This weighted representation is the *cumulative probability function* P_{arr} of ρ_{arr}.

Given a uniformly distributed random variable $\lambda \in [0,1)$, we select an index i so that

$$P_{arr}[i] \leq \lambda \leq P_{arr}[i+1]. \tag{5.4}$$

As visualized in Figure 5.8, this results in i being distributed according to ρ_{arr}.

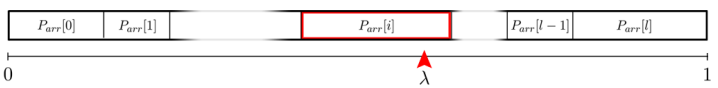


Figure 5.8: An index i is selected according to the position of a random variable λ in the cumulative probability function P_{arr}.

5.5.3 Sampling an Interpolated Function

A set of random data samples generated from a discrete array of size l can only have one distinct data sample per array cell. In order to approximate subcell accuracy, we have to use interpolation.

It is important to note that in this case interpolation is not a trivial problem. We are not looking to re-create any function values of ρ, but are instead searching for a way to manipulate the discrete random data samples generated according to ρ_{arr}. Our goal is matching up the distribution of such generated data samples with the continuous PDF ρ as closely as possible.

An intuitive approach to this is to add an offset using a uniformly distributed random variable $\lambda \in [0,1]$ to the selected index i in order to generate a new index $i_{cont} \in \mathbb{R}$:

$$i_{cont} := i + \left(\lambda - \frac{1}{2}\right).$$

This successfully closes definition gaps in our distribution, allowing us to generate any number of values for each array cell. However, the accuracy of this method can still be improved.

We found that a more precise interpolation can be achieved by emulating a technique that is commonly used in the field of *signal processing*. When recreating an equidistant discrete sampling of a continuous signal, it is known that convolving the signal with a tent filter yields a result akin to linear interpolation.

Consider a triangular PDF t [Sau00]

$$t(x) = \begin{cases} \min(0,\ x+1), & \textbf{if } x \leq 0 \\ \max(1-x,\ 1), & \textbf{else} \end{cases}.$$

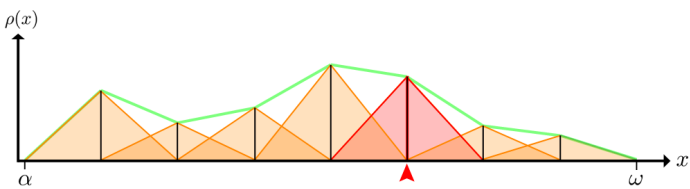


Figure 5.9: Convolution of ρ_{arr} (black) and t (yellow), results in a linearly interpolated function (green).

Using an algorithm to generate random data samples from a triangular distribution [Sau00] we can calculate the interpolated index i_{cont} as:

$$i_{cont} := i + \begin{cases} -1+\sqrt{2\lambda}, & \textbf{if } \lambda \leq 0.5 \\ 1+\sqrt{2(1-\lambda)}, & \textbf{else} \end{cases}, \quad \forall \lambda \in [0,1).$$

In order to calculate the offset of our index, we overlay t onto ρ at the selected index i. This generates a linear probability gradient between i and its neighboring indices. In Figure 5.9 we can see that performing this operation for all indices is equivalent to a convolution of ρ and t. The resulting function, marked in green, is a continuous linear interpolation of ρ_{arr}. In consequence, the distribution described by the randomly generated data samples equals a linearly interpolated sampling of ρ_{arr} (Figure 5.10).

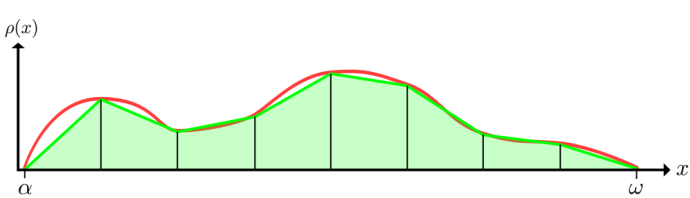


Figure 5.10: Linear interpolation of ρ_{arr} (green) as an approximation of what the user may have intended to draw (red).

This method is not limited to the one-dimensional case but may also be analogously extended into n-dimensions.

5.6 Creating a New Data Set

The creation of a new data set consists of three main steps. The first step is to supply some basic information about the desired data set, e.g. number of dimensions n, number of samples m, number of classes (if any), and a default distribution (e.g. constant, linear or Gaussian) that will be used to model an underlying n-dimensional distribution function. The user may specify data sets with labeling information defining a number of classes. These classes are represented by different colors in the framework. While these classes represent only an additional dimension in the data set, handling them as separate properties allows the user to design the features of each class individually. The user may specify a distinct PDF per class, for each generator object. This allows the user to model clusters with differing labels, as will be shown in Section 5.7.

The second step is to model the desired features into this initial distribution. Features can be designed using the three kinds of generator objects we defined in Section 5.4: one-dimensional PDFs, two-dimensional PDFs, and probability distribution planes (PDPs). These generator objects are defined by one- or two-dimensional probability density functions. The user may specify such functions by selecting statistical distribution parameters and operations provided by the user-interface, as well as manually drawing them on the screen, or loading an image of the appropriate distribution function.

- One-dimensional PDFs: The default distribution function, defined in the first step, is composed of n one-dimensional generator objects. Each of these generator objects initially resembles the supplied default distribution but may later be edited using new distribution parameters, painting, or cropping operations. By editing the one-dimensional PDFs, the user may specify any feature that can be defined in the individual dimensions, e.g. multidimensional clusters.

- Two-dimensional PDFs: Furthermore, our framework allows to design structures in any orthogonal two-dimensional projection of the n-dimensional space using two-dimensional PDFs. The user may select any two distinct dimensions and

attach this generator object to them. The two-dimensional PDF is then defined using a selection of common painting tools (e.g. painting brush, ellipsoids, etc.) or by loading an image containing the desired information. A gray image is computed for the painted or loaded image and these gray values are addressed as a discrete probability function ρ_{arr}, as described in Section 5.5.2). Loading arbitrary images allows users to specify complex features using other imaging tools. Moreover, scatterplot images of existing real data sets can be loaded in order to generate new data sets with similar characteristics as we show in Section 5.7

- PDPs are first specified by positioning them in a three-dimensional subset of the n-dimensional space. Similar to the two-dimensional PDFs, the user may select any three distinct dimensions to place a PDP on. In a three-dimensional view similar to Figure 5.4, the user may adjust the camera to a specific orientation and then simply click and drag the PDP. The four corner points of the PDP are derived from the two points the user defined by clicking and dragging on the screen, and the camera vector is taken as the normal of the plane. To be able to unambiguously calculate these corner points, we further constrain them to form a perfect square from the user's point-of-view. Note that this was done to simplify the process of inserting and editing PDPs, and that the PDP may later be distorted by the user. In addition, the user works with a normalized view of the data set which means that while the PDP may *look* rectangular to the user, it does not necessarily do so in the real data set. As described in Section 5.4.3, there is a two-dimensional PDF mapped onto every PDP. This PDF is defined in an almost identical way to the two-dimensional generator object mentioned above. However, in addition to painting the two-dimensional PDF, the user may also enter scattering- and distortion-factors.

The final step is the evaluation of the generator objects by the algorithm, as described in Section 5.4.4.

5.7 Experiments

In this section we show examples of data sets that were generated using our framework. We generate our examples on an *Intel Core i7 CPU 960 @ 3.20GHz* with 12 GB of RAM. This platform supports modeling synthetic data sets in real-time. Figure 5.11 gives an overview of the average time to generate a sample in relation to the number of dimensions of that sample. It is important to note that, running time is independent of the number of classes of the data set. The system uses approximately 4 kB memory (per class) for each one-dimensional PDF (at least n) and further 2 MB (per class) for each two-dimensional PDF or PDP.

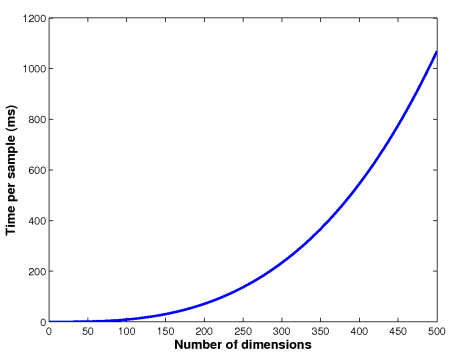
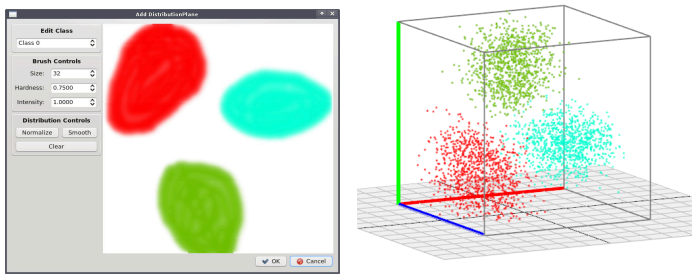


Figure 5.11: Average running times per sample in relation to the number of dimensions.

Our first example is a 10-dimensional data set with three class labels and 3000 samples (1000 per class). Three clusters were defined in the projection (Dim 3 and Dim 5) using our free-hand painting tool (Figure 5.12b). Figure 5.12a shows the created clusters in a 3-dimensional projection including Dims 3, Dim 5, and Dim 2. The remaining dimensions were defined by randomly sampling a normal distribution with mean $\mu = 0$ and variance $\sigma^2 = 1$. Figure 5.13 shows the scatterplot matrix of the generated data set.



(a) Dim 3 and Dim 5 (b) Dim 3, Dim 5 and Dim 2

Figure 5.12: Example of a cluster pattern created with our painting tool. (a) Hand-drawn sketch of the class clusters. (b) 3-dimensional projection including Dim 3, Dim 5 and Dim 2.

The data set we show in Figure 5.14a is the real data set *Abalone* [Nas94], with 4177 instances and 9 attributes that describes the physical measures of abalones. Its main goal is the prediction of the age of an abalone (Dim 8) from its physical measures. Our synthetically generated data set shown in Figure 5.14b was created based on projections of this abalone data set. As mentioned, our framework supports not only painting, but also loading images to define two-dimensional PDFs. Therefore, loading scatterplots of existing real data sets can be used to mimic real features. This example was created by specifying a initial 9-dimensional data set containing 4000 samples. The structure of the new data set was defined by using 8 two-dimensional PDFs, estimated from scatterplots of the original abalone data set. It was created based only on one data set (Abalone), but the framework can also be used to combine scatterplots from different real data set with user-defined structures to create new, unique data sets. The scatterplots marked in red in Figure 5.14a were loaded and smoothed to define two-dimensional PDFs between the respective dimension pairs. Figure 5.15 shows an example of such a scatterplot and its smoothed version. It is worth mentioning that the accuracy of the result (i.e., similarity to the original data) can be improved by using density plots rather than scatterplots. Using our test platform, this data set takes an

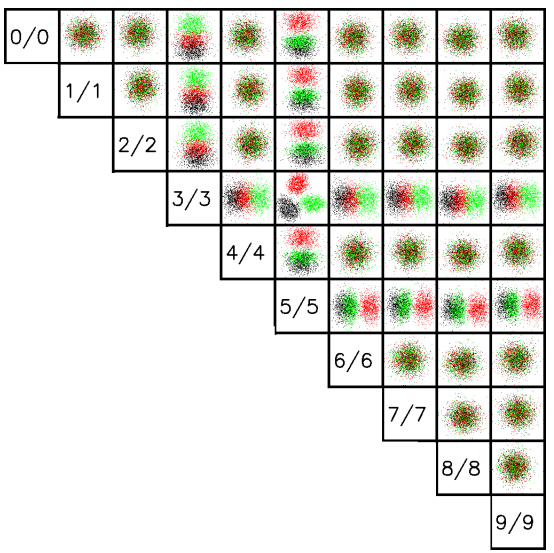
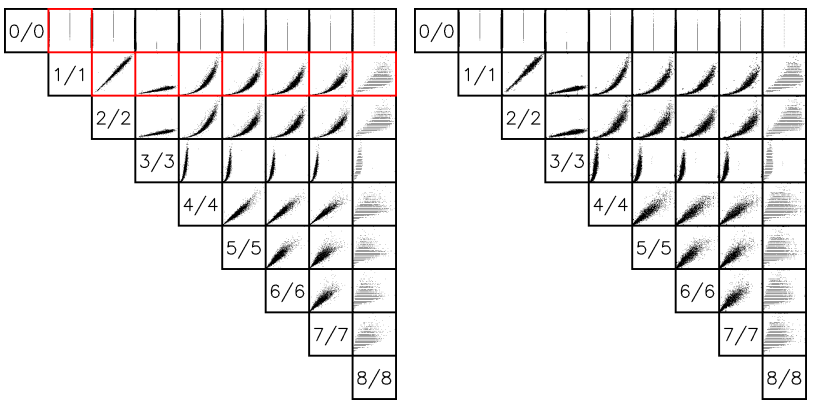


Figure 5.13: Generated data set with 10 dimensions, 1000 samples for each of the three defined classes. Hidden cluster patterns were modeled in two dimensions: Dim 3 and Dim 5. The remaining dimensions were defined by randomly sampling a normal distribution.

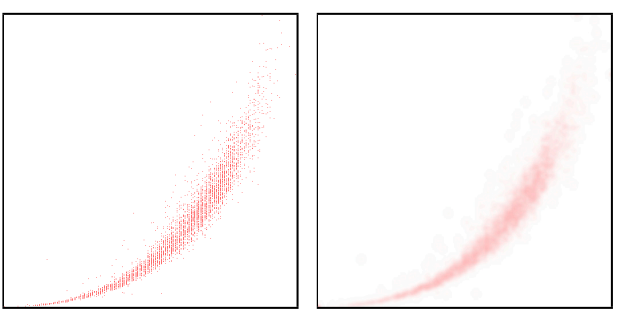
average total of 102.9 milliseconds to generate and it occupies approximately 16 MB of RAM.

Orthographic projection is among the most common visualization tool for multi-dimensional data sets. However, this approach does not always yield satisfying results. As is often the case with natural data, a certain structure that is defined on multiple dimensions may only be visible in non-axis-aligned projections. Using our framework, we are able to easily create such *hidden* structures in our data set. In Figure 5.16, we demonstrate that our system is able to create structures that may not be visible using visualization approaches based on orthographic projections. We show an example data set with 10 dimensions where such a structure is added to a non-orthogonal plane between Dim 3, Dim 8 and Dim 9 (Figure 5.16d). Figures 5.16a, 5.16b, 5.16c show axis-aligned, orthographic projections of the three participating dimensions (Dim 3,



(a) Original data set (b) Synthetically generated data set

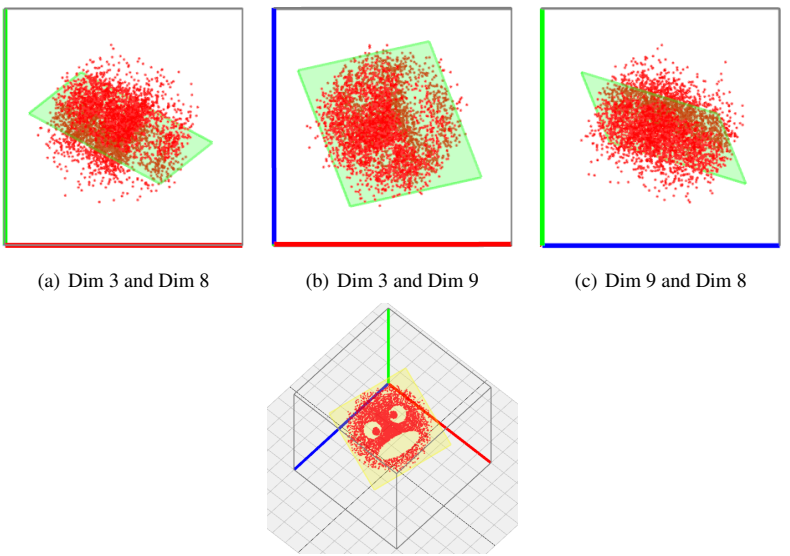
Figure 5.14: Example of a synthetic data set (b), created from projections of a real data set (a). The projections used to model the synthetic data set are marked in red.



(a) Original (b) Smoothed

Figure 5.15: (a) Example of a real scatterplot that is used as a basis for our painting tool. (b) Red intensity represents the probability densities.

Dim 8 and Dim 9) to visualize the data set. However, our *hidden* structure is not recognizable in either of these projections. Figure 5.17 shows a complete view of the data set with a scatterplot matrix. The projections containing the *hidden* structure are marked in red. While it is possible to identify anomalous behavior in those projections,



(a) Dim 3 and Dim 8 (b) Dim 3 and Dim 9 (c) Dim 9 and Dim 8

(d) Hidden structure

Figure 5.16: Data set with 10 dimensions with non-orthogonal structures between Dim 3, Dim 8 and Dim 9 (d). (a), (b) and (c) show axis-aligned, orthographic projections of the three participating dimensions (Dim 3, Dim 8 and Dim 9). (d) shows the hidden structured viewed from the correct, non-orthogonal perspective.

it is not possible to recognize the hidden structure itself.

5.8 Discussion

In this chapter we have presented a novel framework for the synthetic generation of high-dimensional data sets. Specifically, we developed a data generation algorithm based on combining different user inputs. The input data can be real data sets, mathematically defined functions, hand-painted approximated distributions, or a combination of them. Compared to previous approaches our framework is very intuitive and can be used to create data sets with complex structures within minutes. Our primary

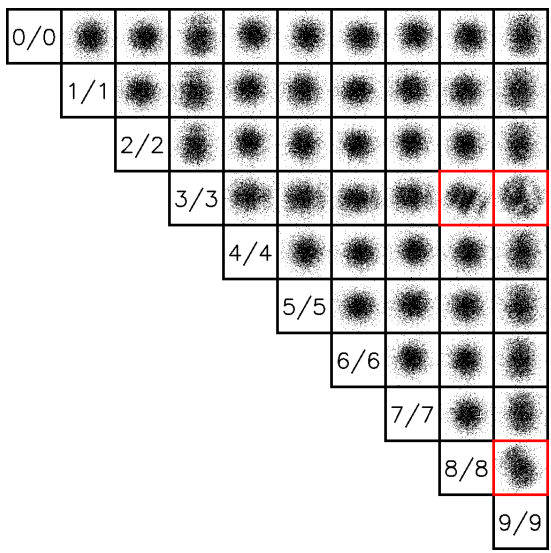


Figure 5.17: Scatterplot matrix of a test data set. Clusters marked in red contribute to our hidden structure.

focus was on granting the user a firm control over the visual aspects of the generated data set, setting our system apart from existing synthetic data generators. This was achieved by designing an intuitive graphical user-interface that allows the user to manually draw trends into a visual representation of the data set. The proposed framework can be used to generate diverse multidimensional trends, including multivariate correlations, clusters, various types of noise, and other statistical structures. Our framework is able to generate axis-independent structures that can be very useful for evaluating visualization approaches that consider non-orthogonal projections of the data set.

The presented contributions are able to aid in and speed up the generation of high-dimensional synthetic data that is needed to evaluate or test a variety of applications.

Chapter 6

Conclusion and Future Directions

In this work, we have presented several automated approaches to support the visual analysis of high-dimensional data sets. First, we developed quality metrics to rank different popular visualization methods including scatterplots, parallel coordinates, radviz, and pixel-oriented displays. This included metrics that are suitable for classified and unclassified data, and for the user tasks of finding correlations and cluster separation. In addition, we introduced the concept of perception-based quality metrics to mimic user opinion while ranking visualizations. Based on the quality metrics, we created new visualization matrices to support the visual analysis of high-dimensional data sets, accompanied by a reordering framework that can facilitate the visual analysis task for matrix-based visualization methods. Finally, we presented a user-friendly sketch-based framework for the generation of synthetic high-dimensional data sets. This framework can be used to generate test cases as proof of concept not only for the approaches presented in this thesis, but also for any visualization tool or algorithm analyses with high-dimensional data sets. The presented methods are able to aid and potentially speed-up the visual exploration process and constitute a step towards the realization of an effective and efficient visual analysis tool for high-dimensional data.

Based on our contributions, various possibilities for future work are conceivable to further improve the visual analysis task. One open issue is to investigate quality metrics that are able to overcome overlapping in the visualizations, e.g. between points

or lines. Overlapping is a very common problem using scatterplots and radviz, but it is even more crucial when using parallel coordinates because the lines can easily cover each other. In such cases, this overlapping could be handled using some level of transparency in the visualizations and adapting the quality metric algorithms. We also intend to investigate higher order ranking functions that are able to cope with more than two dimensional projections. Moreover, perception embeddings for visualizations similar to the approach we presented in Chapter 3 could be used in the future to compare the ranking resulting from of different quality metrics. By mapping the ranking into our perceptual ranking scheme and adopting distances in the embedding as quality values, it could be possible to compare different quality metrics even considering different user tasks. Another direction for future work is the combination of the data generation framework with a suitable visual navigation approach to ease the visualization of the trends in the generation process. Finally, a comparative and qualitative user study can be performed to compare the effectiveness of the quality metrics. An evaluation of the performance of our metrics for scatterplots and parallel coordinates can be found in our paper [TAE+11]. One of our cooperation partners compared our *class density measure*, described in Chapter 2, to user judgment about the best views of a data set in a user study [TBB+10]. A more complete user study for all metrics may prove insightful.

By the time this thesis is written, further research inspired by the approaches we presented has been published. Ferdosi et al. [FR11] propose a new dimension reordering algorithm based on subspace analysis to reduce clutter in parallel coordinates. They compare their method with our *hough space measure* described in Chapter 2 and observe that their method could better support the visualization of multidimensional clusters. Related to our perception-based quality metric presented in Chapter 3, Scherer et al. [SBS11] propose a method for content-based retrieval and exploration in multidimensional data sets, where a set of regression models are used to search for projections of the data set with similar models. Heinrich and Weiskopf [HW13] present a taxonomy for parallel coordinates methods. Related to our parallel coordinates matrix (Chapter 4), Heinrich et al. [HSW12] propose another parallel coordinates matrix

that displays all pairwise relations without redundancy. Bertini et al. [BTK11] present an overview and systematization of existing quality metrics, and new quality metrics have been proposed by [DWA12] to characterize visual patterns of the low-dimensional projections of multidimensional data sets that constitute feature subspaces. Finally, for synthetic data generation, a new approach, similar to our framework presented in Chapter 5 was proposed by Wang et al. [WRM13] that supports the creation of data sets using parallel coordinates and an N-D polygon to navigate in the multidimensional space.

6. CONCLUSION AND FUTURE DIRECTIONS

References

[ABCC07] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook. *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton University Press, 2007.

[ABK98] M. Ankerst, S. Berchtold, and D. A. Keim. Similarity clustering of dimensions for an enhanced visualization of multidimensional data. In *IEEE Symposium on Information Visualization (InfoVis)*, pages 52–60, 1998.

[ACDV94] S. Aeberhard, D. Coomans, and O. De Vel. Comparative-analysis of statistical pattern-recognition methods in high-dimensional settings. pattern recognition. In *IEEE Signal Processing Workshop on Higher Order Statistics*, pages 14–16, 1994.

[AEL+09] G. Albuquerque, M. Eisemann, D. J. Lehmann, H. Theisel, and M. Magnor. Quality-based visualization matrices. In *Vision, Modeling and Visualization (VMV)*, pages 341–349, 2009.

[AEL+10] G. Albuquerque, M. Eisemann, D. J. Lehmann, H. Theisel, and M. Magnor. Improving the visual analysis of high-dimensional datasets using quality measures. In *IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 19–26, 2010.

[AEM11] G. Albuquerque, M. Eisemann, and M. Magnor. Perception-based visual quality measures. In *IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 13–20, 2011.

[AIS93] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD International Conference on Management of Data*, pages 207–216, 1993.

[ALM11] G. Albuquerque, T. Löwe, and M. Magnor. Synthetic generation of high-dimensional datasets. *IEEE Transactions on Visualization and Computer Graphics (TVCG, Proc. Visualization / InfoVis)*, 17(12):2317–2324, 2011.

[AMN+94] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Wu. An optimal algorithm for approximate nearest neighbor searching. In *ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 573–582, 1994.

[AMS97] K. Ali, S. Manganaris, and R. Srikant. Partial classification using association rules. In *International conference on Knowledge Discovery and Data mining (KDD)*, pages 115–118, 1997.

[Asi85] D. Asimov. The grand tour: a tool for viewing multidimensional data. *Journal on Scientific and Statistical Computing*, 6(1):128–143, 1985.

[AWC+07] S. Agarwal, J. Wills, L. Cayton, G. Lanckriet, D. Kriegman, and S. Belongie. Generalized non-metric multidimensional scaling. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1–8, 2007.

[BI07] D. Borland and R. Taylor II. Rainbow color map (still) considered harmful. *IEEE Computer Graphics and Applications*, 27(2):14–17, 2007.

[BL13] K. Bache and M. Lichman. University of california irvine (UCI) machine learning repository, 2013.

[BSMM01] I. N. Bronstein, K. A. Semendjajew, G. Musiol, and H. Mühlig. *Taschenbuch der Mathematik*. Verlag Harri Deutsch, Frankfurt am Main, 2001.

[BTK11] E. Bertini, A. Tatu, and D. Keim. Quality metrics in high-dimensional data visualization: An overview and systematization. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2203–2212, 2011.

[Bul79] M. G. Bulmer. *Principles of statistics*. Dover Publications, 2 edition, 1979.

[CBCH95] D. Cook, A. Buja, J. Cabreta, and C. Hurley. Grand tour and projection pursuit. *Journal of Computational and Statistical Computing*, 4(3):155–172, 1995.

[Cha83] J. M. Chambers. *Graphical methods for data analysis*. Chapman & Hall statistics series. Wadsworth International Group, 1983.

[Cle84] W. S. Cleveland. Graphical methods for data presentation: Full scale breaks, dot charts, and multibased logging. *The American Statistician*, 38(4):270–280, 1984.

[Dal02] P. Dalgaard. *Introductory Statistics with R*. Springer, 2002.

[Dav88] H. A. David. *The Method of Paired Comparisons*. Chapman and Hall, London, 2 edition, 1988.

[DK10] A. Dasgupta and R. Kosara. Pargnostics: Screen-space metrics for parallel coordinates. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 16:1017–1026, 2010.

[DO91] R. A. DeMillo and A. J. Offutt. Constraint-based automatic test data generation. *IEEE Transactions on Software Engineering*, 17(9):900–910, 1991.

[Dup26] C. Dupin. Carte figurative de l´ instruction populaire de la france, 1826.

[DWA12] T. N. Dang, L. Wilkinson, and A. Anand. Visual pattern discovery using random projections. pages 43–52, 2012.

[EAM11] M. Eisemann, G. Albuquerque, and M. Magnor. Data driven color mapping. In *EuroVA: International Workshop on Visual Analytics*, pages 5–8, 2011.

[FFT75] M. A. Fisherkeller, J. H. Friedman, and J. W. Tukey. PRIM-9: An interactive multi-dimensional data display and analysis system. In *ACM Pacific*, pages 140–145, 1975.

[FR11] B. J. Ferdosi and J. B. T. M. Roerdink. Visualizing high-dimensional structures by dimension ordering and filtering using subspace analysis. In *Joint Eurographics / IEEE VGTC Conference on Visualization (EuroVis)*, pages 1121–1130, 2011.

[Fri87] J. H. Friedman. Exploratory projection pursuit. *Journal of the American Statistical Association*, 82(397):249–266, 1987.

[FT74] J. H. Friedman and J. W. Tukey. A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on Computers*, C-23(9):881–890, 1974.

[GCML06] D. Guo, J. Chen, A. M. MacEachren, and K. Liao. A visualization system for space-time and multivariate patterns (vis-stamp). *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 12(6):1461–1474, 2006.

[GHLP01] G. Grinstein, P. Hoffman, S. J. Laskowski, and R. M. Pickett. Benchmark development for the evaluation of visualization for data mining. In *Information Visualization in Data Mining and Knowledge Discovery*, pages 129–176, 2001.

[GSF77] T. Gonzalez, S. Sahni, and W. R. Franta. An efficient algorithm for the kolmogorov-smirnov and lilliefors tests. *Transactions on Mathematical Software (TOMS)*, 3(1):60–64, 1977.

[Guo03] D. Guo. Coordinating computational and visual approaches for interactive feature selection and multivariate clustering. *Information Visualization*, 2(4):232–246, 2003.

[GV04] H. Guo and H. L. Viktor. Learning from imbalanced data sets with boosting and data generation: the databoost-im approach. *ACM SIGKDD Explorations Newsletter*, 6(1):30–39, 2004.

[GW06] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. 3 edition, 2006.

[Har75] J. A. Hartigan. Printer graphics for clustering. *Journal of Statistical Computation and Simulation*, 4(3):187–273, 1975.

[HD79] F. Hartwig and B. E. Dearing. *Exploratory Data Analysis*. SAGE Publications, 1979.

[HGM+97] P. Hoffman, G. Grinstein, K. Marx, I. Grosse, and E. Stanley. DNA visual and analytic data mining. In *IEEE Visualization (VIS)*, pages 437–ff., 1997.

[HGP99] P. Hoffman, G. Grinstein, and D. Pinkney. Dimensional anchors: a graphic primitive for multidimensional multivariate information visualizations. In *Workshop on New Paradigms in Information Visualization and Manipulation (NPIVM)*, pages 9–16, 1999.

[HNR68] P. N. Hart, N. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics,*, 4(2):100–107, 1968.

[Hou62] P. Hough. Method and means for recognizing complex patterns. U.S. Patent 3.069.654, 1962.

[HSW12] J. Heinrich, J. Stasko, and D. Weiskopf. The parallel coordinates matrix. In *EuroVis - Short Papers*, pages 37–41. Eurographics Association, 2012.

[Hub85] P. J. Huber. Projection pursuit. *The Annals of Statistics*, 13(2):435–475, 1985.

[HW13] J. Heinrich and D. Weiskopf. State of the art of parallel coordinates. In *STAR Proceedings of Eurographics*, pages 95–116, 2013.

[Ins85] A. Inselberg. The plane with parallel coordinates. *The Visual Computer*, 1(4):69–91, 1985.

[JJ09] S. Johansson and J. Johansson. Interactive dimensionality reduction through user-defined combinations of quality metrics. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 15(6):993–1000, 2009.

[JLR+06] D.R Jeske, P.J Lin, C. Rendon, R. Xiao, and B. Samadi. Synthetic data generation capabilties for testing data mining tools. pages 1–6, 2006.

[JSL+05] D.R. Jeske, B. Samadi, P.J. Lin, L. Ye, S. Cox, R. Xiao, T. Younglove, M. Ly, D. Holt, and R. Rich. Generation of synthetic data sets for evaluating the accuracy of knowledge discovery systems. In *ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pages 756–762, 2005.

[KAK95] D. A. Keim, M. Ankerst, and H.-P. Kriegel. Recursive pattern: A technique for visualizing very large amounts of data. In *IEEE Visualization (VIS)*, pages 279–286, 1995.

[KAS04] D. A. Keim, M. Ankerst, and M. Sips. *Visual Data-Mining Techniques*, pages 813–825. Kolam Publishing, 2004.

[KC03] Y. Koren and L. Carmel. Visualization of labeled data using linear transformations. pages 121–128, 2003.

[Kei00] D. A. Keim. Designing pixel-oriented visualization techniques: Theory and applications. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 6(1):59–78, 2000.

[Kei02] D. A. Keim. Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 8(1):1–8, 2002.

[KHDH02] D. A. Keim, M. C. Hao, U. Dayal, and M. Hsu. Pixel bar charts: A visualization technique for very large multi-attribute data sets. *Information Visualization*, 1(1):20–34, 2002.

[KKEM10] D. A. Keim, J. Kohlhammer, G. Ellis, and F. Mansmann. *Mastering The Information Age - Solving Problems with Visual Analytics*. Eurographics, 2010.

[KMSZ06] D. A. Keim, F. Mansmann, J. Schneidewind, and H. Ziegler. Challenges in visual data analysis. In *IEEE Symposium on Information Visualization (InfoVis)*, pages 9–16, 2006.

[Llo82] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[LMH+09] M. A. Little, P. E. McSharry, E. J. Hunter, J. L. Spielman, and L. O. Ramig. Suitability of dysphonia measurements for telemonitoring of parkinson's disease. *IEEE Trans. Biomed. Engineering*, 56(4):1015–1022, 2009.

[LMR+07] M. Little, P. McSharry, S. Roberts, D. Costello, and I. Moroz. Exploiting nonlinear recurrence and fractal scaling properties for voice disorder detection. *BioMedical Engineering OnLine*, 6(1):1–23, June 2007.

[Mac67] J. B. Macqueen. Some methods for classification and analysis of multivariate observations. In *Berkeley Symposium on Math, Statistics, and Probability*, pages 281–297, 1967.

[MDK10] T. May, J. Davey, and J. Kohlhammer. Combining details of the chi-square goodness-of-fit test with multivariate data visualization. In *International Symposium on Visual Analytics Science and Technology (EuroVast)*, pages 45–50, 2010.

[Mic89] T. Micceri. The unicorn, the normal curve, and other improbable creatures. *Psychological Bulletin*, 105(1):156–166, 1989.

[MMSW97] C. C. Michael, G. E. McGraw, M. A. Schatz, and C. C. Walton. Genetic algorithms for dynamic test data generation. In *IEEE International Conference Automated Software Engineering*, pages 307–308, 1997.

[Nas94] W. J. Nash. *The Population biology of abalone (Haliotis species) in Tasmania. 1, Blacklip abalone (H. rubra) from the north coast and the islands of Bass Strait*. Sea Fisheries Division, Dept. of Primary Industry and Fisheries, Tasmania, Hobart :, 1994.

[NJW01] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, pages 849–856, 2001.

[Nv09a] L. Nováková and O. Štěpánková. Radviz and identification of clusters in multidimensional data. In *IEEE Information Visualisation (InfoVis)*, pages 104–109, 2009.

[Nv09b] L. Nováková and O. Štěpánková. Visualization of trends using radviz. In *International Symposium on Foundations of Intelligent Systems (ISMIS)*, pages 56–65, 2009.

[PHP99] R. P. Pargas, M. J. Harrold, and R. R. Peck. Test-data generation using genetic algorithms. *Software Testing, Verification and Reliability*, 9(4):263–282, 1999.

[PWR04] W. Peng, M. O. Ward, and E. A. Rundensteiner. Clutter reduction in multidimensional data visualization using dimension reordering. In *IEEE Information Visualisation (InfoVis)*, pages 89–96, 2004.

[Rei02] J. P. Reiter. Satisfying disclosure restrictions with synthetic data sets. *Journal of Official Statistics - Stockholm*, 18(4):531–544, 2002.

[RW95] P. Russell and A. Williams. *The Nutrition and Health Dictionary*. Chapman and Hall, 1995.

[Sau00] R. Saucier. Computer generation of statistical distributions. Technical Report 4ARL-TR-2168, Army Research Laboratory, 2000.

[SBS11] M. Scherer, J. Bernard, and T. Schreck. Retrieval and exploratory search in multivariate research data repositories using regressional features. In *International ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pages 363–372, 2011.

[SGM08] J. Sharko, G. Grinstein, and K. A. Marx. Vectorized radviz and its application to multiple cluster datasets. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 14(6):1444–1427, 2008.

[Shn96] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *IEEE Symposium on Visual Languages*, pages 336–343, 1996.

[SNLH09] M. Sips, B. Neubert, J. P. Lewis, and P. Hanrahan. Selecting good views of high-dimensional data using class consistency. *Computer Graphics Forum (Proc. EuroVis 2009)*, 28(3):831–838, 2009.

[SS05] J. Seo and B. Shneiderman. A rank-by-feature framework for interactive exploration of multidinemsional data. *Information Visualization*, 4(2):96–113, 2005.

[SSK06] J. Schneidewind, M. Sips, and D. A. Keim. Pixnostics: Towards measuring the value of visualization. pages 199–206, 2006.

[STBC03] D. F. Swayne, D. Temple Lang, A. Buja, and D. Cook. GGobi: evolving from XGobi into an extensible framework for interactive data visualization. *Computational Statistics & Data Analysis*, 43(4):423–444, 2003.

[SWM93] W. N. Street, W. H. Wolberg, and O. L. Mangasarian. Nuclear feature extraction for breast tumor diagnosis. pages 861–870, 1993.

[TAE+09] A. Tatu, G. Albuquerque, M. Eisemann, J. Schneidewind, H. Theisel, M. Magnor, and D. Keim. Combining automated analysis and visualization techniques for effective exploration of high-dimensional data. pages 59–66, 2009. Won the SPP Collaboration Award in the DFG priority program on Scalable Visual Analytics (SPP 1335).

[TAE+11] A. Tatu, G. Albuquerque, M. Eisemann, P. Bak, H. Theisel, M. Magnor, and D. Keim. Automated analytical methods to support visual exploration of high-dimensional data. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 17(5):584–597, 2011.

REFERENCES

[TBB+10] A. Tatu, P. Bak, E. Bertini, D. Keim, and J. Schneidewind. Visual quality metrics and human perception: an initial study on 2d projections of large multidimensional data. In *International Conference on Advanced Visual Interfaces (AVI)*, pages 49–56, 2010.

[TP91] M. Turk and A. P. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[TT85] J. Tukey and P. Tukey. Computing graphics and exploratory data analysis: An introduction. In *Annual Conference and Exposition: Computer Graphics. Nat. Computer Graphics Assoc.*, 1985.

[WAG05] L. Wilkinson, A. Anand, and R. Grossman. Graph-theoretic scagnostics. In *IEEE Symposium on Information Visualization (InfoVis)*, pages 157–164, 2005.

[WAKB09] J. Wills, S. Agarwal, D. Kriegman, and S. Belongie. Toward a perceptual space for gloss. *ACM Transactions on Graphics (TOG)*, 28:103:1–103:15, 2009.

[War94] M. O. Ward. Xmdvtool: Integrating multiple methods for visualizing multivariate data. In *IEEE Symposium on Information Visualization (InfoVis)*, pages 326–333, 1994.

[Wat05] M. Wattenberg. A note on space-filling visualizations and space-filling curves. In *IEEE Symposium on Information Visualization (InfoVis)*, pages 181–186, 2005.

[Weg90] E. J. Wegman. Hyperdimensional data analysis using parallel coordinates. *Journal of the American Statistical Association*, 85(411):664–675, 1990.

[WLKT09] L.-Y. Wei, S. Lefebvre, V. Kwatra, and G. Turk. State of the art in example-based texture synthesis. In *STAR Proceedings of Eurographics*, pages 93–117, 2009.

[Wri38] J. K. Wright. Problems in population mapping. *Notes on Statistical Mapping With Special Reference to the Mapping of Population Phenomena*, pages 1–18, 1938.

[WRM13] B. Wang, P. Ruchikachorn, and K. Mueller. SketchPadN-D: WYDIWYG sculpting and editing in high-dimensional space. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 19(12):2060–2069, 2013.

[YWRH03] J. Yang, M. Ward, E. Rundensteiner, and S. Huang. Visual hierarchical dimension reduction for exploration of high dimensional datasets. In *Symposium on Data Visualisation (VISSYM)*, pages 19–28, 2003.

[ZF08] K. Zhang and W. Fan. Forecasting skewed biased stochastic ozone days: analyses, solutions and beyond. *Knowledge and Information Systems*, 14(3):299–326, 2008.

[ZNLG94] J. Zupan, M. Novic, X. Li, and J. Gasteiger. Classification of multicomponent analytical data of olive oils using different neural networks. *Analytica Chimica Acta*, 292(1):219–234, 1994.